DOUBLE ISSUE

Volume 1
Number 3

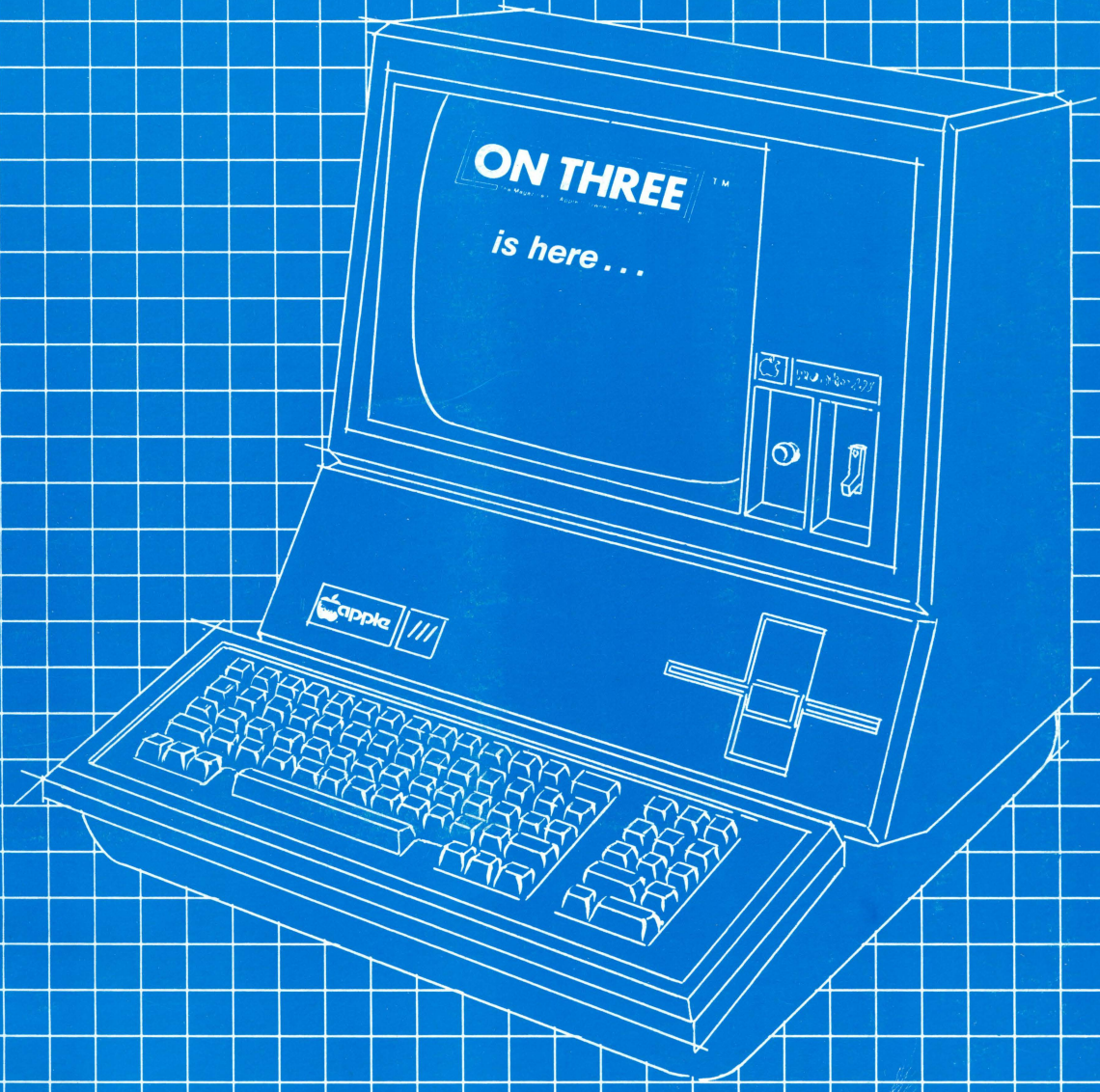
ON THREE T.M.

The Magazine For Apple III Owners and Users

$3.00

April-May
1983

- Disk Pak 3: Catalog your DOS diskettes
- SpeedBoot (Un) Lock?
- Changing the Character of the Apple Dot Matrix Printer
- WPL Tutorial begins
- REVIEW ON: Everything
- Plus much more!



ON THREE Presents . . .

μ-SCI

MICRO-SCI

Micro-Sci Disk Drives

Every once in a while a product appears that is so good **ON THREE** decides to offer it for sale to our readers. ***The Micro-Sci*** line of disk drives (and the ***Gameport ///***) are the first of these superior type products. Byte for byte, these drives offer greater speed and more value than any comparable drive on the market today. If you are looking into purchasing an external disk drive for your ///, **ON THREE** encourages you to look into this fantastic product line.

Expanding disk storage on the Apple /// can be an expensive proposition.

But ***Micro-Sci*** has a better proposition for you, because our disk drives for the Apple /// give you greater capacity and performance for every dollar spent.

And there are no compatibility problems. The A3 is a direct replacement for Disk /// drives, and the 70-track A73 and 140-track A143 are supplied with a driver that is easily added to the SOS driver module, affording extra storage and fast seek rates for all of the programs that run under SOS.

Talk about compatible! All three are the same size as your built-in drive and they use the same diskettes!

Are all of your slots full? Don't worry, these drives plug right into the back of your /// and they don't need a power cord! Up to three extra disk drives can be daisy-chained and they can be mixed in any combination of Disk ///, A3, A73 or A143.

The A3 offers identical capacity to the Disk /// and is an excellent choice for a second disk compatibility in the Apple][emulation mode.

At 286 KBytes, the A73 has double the capacity of the Disk /// while the ***A143 packs 572 KBytes*** of data onto a diskette. With over ***half a megabyte*** of storage space, the A143 makes a truly viable backup device for the Profile Hard Disk.

With that large a capacity, many people find that they ***don't need a hard disk!*** Since up to three A143's can be used with your ///, you can have over one and three quarter megabytes of data on-line at all times!

ON THREE is pleased to announce the following low, low prices on these great disk drives.

	A3	**A73**	**A143**
Suggested List Price:	$379	$529	$659
ON THREE Price:	$299	$409	$509
Savings	$80	$120	$150

To order, use the attached envelope and add $6.50 for postage and handling for each drive ordered. Please allow four weeks for delivery.

Gameport ///

You don't have to be chained to your job, and neither does your Apple ///. After the working day is done, release your computer into the exhilarating world of adventure and challenge with a ***Gameport ///*** from ***Micro-Sci.*** The new Gameport /// game controller adapter lets you use game paddles, joysticks and all your favorite Apple][amusement packages with your Apple /// computer. The Gameport /// is easy to use and simple to install - your only challenge is to conquer the invaders!

The Gameport ///

- Allows all games written for the Apple][to be used on the Apple ///.
- Works with all Apple][game paddles and joysticks.
- Allows programs which require a game I/O protection key to run in Apple][emulation mode.
- Can be installed in any slot.
- Does not interfere with the normal operation of the Apple ///.
- Package includes: Gameport /// board, Apple][Emulation Modification Diskette and complete, easy-to-follow instructions (Apple][game controllers not included).

ON THREE proudly sells the ***Gameport ///*** by ***Micro-Sci.*** For only $59.95 you can now get the best that the Apple /// and the Apple][has to offer. That's $15 off the suggested list price so don't be left out, place your order today! Please use the attached envelope for ordering and remember to add $2.50 for postage and handling.

ON THREE

/// TABLE OF CONTENTS ///

The Editor's Block:
Bob Consorti 2

Ask THREE:
(Letters to the Editor) 3

/// FEATURES ///

DOS File List:
Bob Consorti 10

Assembling (ON) the *III*:
Martin Nichols 17

Changing the Character of the Apple Dot Matrix Printer:
Timothy S. Smith 13

Products Received:
Some new, some old — all good 19

/// DEPARTMENTS ///

WPL Tutorial:
Use it!
Bob Consorti 20

REVIEW ON: Everything
Pkaso, Quick & Easy Data Master, ProFile & Backup ///, Critical Path Scheduling
Bob Consorti 24

/// to the Max:
Making those horses run!
Al Evans 22

Three Shorts — Fini!
More graphic demos for your ///.
Devin Sexson 32

Micro-Sci disk drives, Gameport /// ..... Inside Front Cover
Lazarus ///, ON THREE O'Clock, Disk Of the Month #1 & #2 ..... Inside Back Cover

Next Month in ON THREE

Disk Verify+ . . . Formatting disks from Basic . . . Keyboard Change . . . Color fix . . . Paddle fix . . . Reviews: GL-PLUS, MICROTEK Dumpling, VersaForm, Stock Portfolio System . . . tutorials . . . and more!

April-May, 1983
Volume 1 Number 3

Editor/Publisher:
Bob Consorti

Managing Consultant:
Joseph Consorti

Cover Design and Artwork:
William V. Padula
Cranford, New Jersey

Interior Artwork:
Virginia Carol

Typesetting Services:
The Typesetting Company
Van Nuys, California

Printing Services:
Ojai Printing & Publishing Company
Ojai, California

ON THREE - **THE** Reference Source for the Apple /// is published somewhat monthly by ***ON THREE,*** P.O. Box 3825, Ventura, California 93006.

For a copy of our Author Guidelines, please send a self-addressed stamped envelope (20 cents) to the above address.

Subscription information: U.S. - $30 for 12 issues. For first class mail, please remit an extra $10.

All foreign subscriptions should include additional postage in the following amounts:

$8 for Canada and Mexico
$12 for Central America - Caribbean
$16 for South America - Europe - Africa
$19 for Asia - Pacific Islands - Australia

Funds should be remitted in either U.S. dollars drawn on a U.S. bank, International Money Order, or by a direct bank draft.

Group Rates are as follows:

2 - 9 members, $28 per subscription
10-49 members, $25 per subscription
50 + members, $22 per subscription

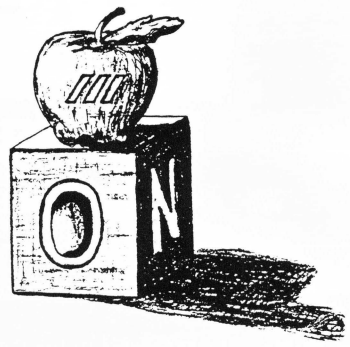
Group purchases must have one mailing address.

Return postage must accompany all manuscripts, drawings, and diskettes submitted if they are to be returned, and no responsibility can be assumed for unsolicited materials.

All letters sent to ***ON THREE*** will be treated as unconditionally assigned for publication and are subject to ***ON THREE's*** right to edit and comment editorially.

Dealer inquiries are welcomed. Please write to the above address for volume pricing and terms.

The Editor's Block:



Bob Consorti

Well, we're back again! Thanks for all of your support, I don't know how we would do it without your help. Yet, even with the tremendous support Apple /// users have given us, putting together a quality publication is both hard work and time-consuming. For those of you wondering, we will go to a monthly format as soon as possible.

The biggest problem we have encountered to date is in getting enough articles to fill up an issue. Since I'm sure we all want to see an issue a month very soon, I am requesting that if you can submit an article, please do. We need your help!

Now that my blood pressure has gone down a bit, let me introduce you to **ON THREE'S** latest products. Joining the ever popular **ON THREE O'Clock** and the **DOM #1** are SIX new products!

Are you looking into buying another disk drive? Have we got a deal for you! **ON THREE** is proud to announce that we are now selling the **Micro-Sci** line of disk drives for the Apple ///. Very easy to install and use, these drives don't use up a slot and are completely compatible with all Apple /// programs. The **A143** packs **572K** Bytes of data onto a diskette. That's **over half a megabyte!**

If you have a ProFile and are tired of wasting two hours and 40 floppy disks to back it up, **Micro-Sci** disk drives are just what you've been waiting for. And if you don't have a hard disk but feel limited by the standard old 140K disk drives, try out the **A143** - you'll love it!

Sounds pretty good, right? Well the bottom line is the price. **Only $509** for over half a megabyte of fast data storage. If you are just looking for a second 140K floppy disk drive, the **A3** is a direct replacement for the Disk /// and is priced at **only $299.** Last but not least, the **A73** provides double the storage of the standard Apple drive and costs just **$409.**

A more complete description of these three disk drives is found on the inside front cover. Check them out, you'll be glad you did!

Three down, three to go! The next new product is the **Gameport ///** by **Micro-Sci.** It lets you use game paddles, joysticks and ALL your favorite Apple][amusement software with your Apple ///. Easy to install and use, **ON THREE** offers the **Gameport ///** for only $59.95. See the inside front cover for more details.

The next to last item is the **second Disk Of the Month!** Following our commitment to bringing you the best products at the lowest possible price, it contains all the useful programs in this month's issue, and is priced at **only $9.95.** You can now list Apple][files from your /// as well as changing the characters of your Apple DMP or Prowriter. What a buy!

Last and certainly not least is the most important Apple /// utility that will ever come out. I'm talking about none other than **Lazarus ///,** the package that will recover the files you accidentally deleted.

Following Murphy's Law, the one file you accidentally delete will be the only one you haven't ever backed up. With **Lazarus ///** you will never again have to retype months and months of important data. Just insert the diskette with the deleted files and **Lazarus ///** will recover it.

Completely user friendly, this program has on-line help and tutorial screens to give help to you novices out there. It even works with the ProFile and other disk drives! If you place your order today, you can get it for the pre-introductory price of **$24.95.** Similar programs for the Apple][have been sold for over $100, so I think you'll agree with me when I say it's a steal! See the inside back cover for ordering details.

Switching now to a more complicated subject, many people have written to me saying that Apple has not been very supportive of the /// in the past. Most everyone also seems to think that Apple is going to drop the /// completely. While even Apple will admit it's having problems with the ///, they are not going to 'drop-it'.

I won't try and make excuses for Apple, that's not my job. Let me just say that they are aware that there is a serious problem and they are attempting to correct it. People ask me how and when, but I can't answer that one. In my present position I have access to some information about future Apple /// offerings. However, I am bound by non-disclosure agreements so I can't say exactly what. But they are definitely commited to the Apple ///'s future, so don't worry.

Now it's time to turn your back on all the problems of the world and look inside the latest issue of **ON THREE.** One of the things that **ON THREE** does is answer your questions. Look over the Letters to the Editor this month. There is a heck of a lot of information on those pages that YOU can use. Talk about information, look over our review section this month. Now that's a lot!

This month Timothy Smith shows how to change the characters that your Apple DMP or Prowriter prints with. A very nice utility program is included that lets your DMP print with any of the screen fonts. Now your printer can use the fancy Gothic characters, or the modern Byte character set. Any Apple /// screen font can be used!

For those of you who want it, with the information and programs in the article DOS File List, you can list the files on any Apple][DOS diskette. Just another handy utility from the folks at ON THREE. Also included are some handy invokable routines by Martin Nichols.

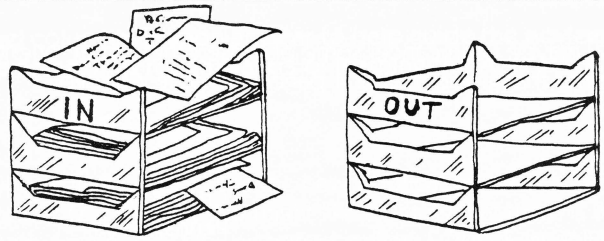
Al Evans is back with '/// to the Max'. In this article he shows just what makes the horses run! The long awaited WPL tutorial commences this month. Here you will learn just what WPL can do for you! In the next issue, Earl Curlson and Louis Hanson will resume their popular tutorials.

Next time I'll show you a program to do a complete verification of your disks and more. Martin Nichols will bring you a routine to format disks in Basic, and maybe even a special on Pascal! Reviews, reviews and more reviews. And would you beleive it?, two hardware modifications for your ///. One to allow color in the emulation mode and another to fix the game paddle problem. We will even squeeze in an article on spreadsheeting - from Basic!

Before I leave you, let me put out a call to all Apple /// users who think they know their machine: **ON THREE** is presently setting up a **HOT-LINE** service to help Apple /// users overcome their problems. If you know enough to answer questions, please write or give us a call. In a very short time I hope to have enough of our readers willing to spend a little time answering questions that any Apple /// owner could dial a relatively local number and get immediate help.

The Editor's Block continued on page 9.

Ask THREE: (Letters to the Editor)



Dear Bob,

What a GREAT publication — **ON THREE.** Initially thought you were spoofing about not having enough time to type in the programs . . . thus buy the disks. But you're right! These are MUST haves for the wealth of information contained thereon. The mini-tutorials are sorely needed and are the best yet to come out specifically for the ///.

We have a situation which could be addressed to you or Louis Hanson for a clue to overcome. During the past two years we've worked up some 37 integrated programs producing a neat package "Municipal Tax Management Systems" for the local tax collector which produce the tax bills, keep track of ownership changes, maintain the assessor's records, post payments, late charges, interest penalties, etc. We've marketed two sets so far, and are receiving inquires from owners of other brands of computers. As a result, we're in the process of converting the programs from the originally written Basic to Pascal. The data base on which these were written is a 2500 account BasicData file. We're wondering if you know of a utility to convert Basdata files to Pasdata files so that we do not have to manually type the files all in again. Let me tell you, THAT is some chore!

Interesting clock you have . . . considerably different than our Thunderclock. With the Thunderclock we've learned to date-stamp a program with a little text file routine, though it takes several seconds to apply. Haven't tried to access it in Pascal yet so can't tell whether it can be done. Any thoughts on this will certainly save us considerable experimental time.

Your fledgling "ON THREE" is moving right to the heart of getting the best out of the ///. It deserves ALL Three'ers strong and continuing support. You certainly have ours.

Very Truly Yours,

Coville Woodburn
New Hampshire

Dear Mr. Woodburn,

I'm very glad that you are enjoying the magazine, we are doing our best to fill it with a wealth of information. I'm also pleased to see that you like the tutorials. We are planning quite a few more in the near future.

On the subject of the Basic data files, read pages 213 through 216 in the Pascal Programmer's Manual, Volume 1. Here the description of the way Pascal handles different types of files should help you find a solution to the conversion problem.

You see, to convert a Basic text or data file to a Pascal format, you must have planned out a file data and record structure for the conversion. You can read in a record from the Basic data file into the correct record format and then write it out in your defined data structure format.

The ON THREE O'Clock interfaces with SOS directly, no special drivers are needed and no complicated date-stamp routines are neccessary. Whenever you first save a file and whenever you modify it, these times and dates are automatically put into the file's directory entry.

Your Basic programs can also use the reserved words 'DATE$' and 'TIME$' to provide you with an up to the second read-out on what time it is. From Pascal you can access the internal date and time by using the Applestuff Unit as described on pages 50-52 in the Pascal Programmer's Manual, Volume 2.

Thank you again for your support.

Dear ON THREE,

At last, a magazine and a group of people dedicated to in-depth support of the fantastic Apple ///! Enclosed is my check for a one-year subscription. Please be sure to notify me when the reprint of the January issue is available—I definitely want to obtain a copy.

All this enthusiasm, and I don't even have my Apple /// yet! I expect to get it soon, though—perhaps this week. My enthusiasm for **ON THREE** is because up to now I've been frustrated with how slowly support for the /// has been developing. But the /// is such a beautiful machine that I decided to "rough-it" and buy it instead of an Apple //e. Now I find your organization is available to help me over the rough spots. I have already passed knowledge of your existence on to others and I will work hard at this in the future because I agree the more support we can get, the more it will benefit us all. (Maybe I can talk the local stores into a list of their Apple /// buyers so I can contact them myself.) I also hope to make contributions to your magazine in the future.

Incidentally, I am pleased to see so much emphasis on Pascal in **ON THREE,** since that is my preferred language and one of the reasons I'm buying an Apple ///.

So sign me up, and let me know when the January reprint is ready.

Sincerely,

Hobart S. Cable, II
Ohio

Dear Mr. Cable,

Your enthusiasm is great! ON THREE is here to do just what you say - help the Apple /// user over the rough spots. I'm very glad we can help, so if you have any questions, please do write again.

Gentlemen,

Finally there appears information on the Apple ///! Please rush subscription information plus details on how I may order all back issues.

Bless you for coming to the aid of us Apple /// users left completely out in the cold by Apple Computer, Inc.

Now maybe I can cancel some of those other Apple oriented magazines that give only 3 or 4 pages of Apple /// info.

Thanks,

Charles Miller
Pennsylvania

Dear Mr. Miller,

I'm glad that we can help you. One of the big reasons I started ON THREE was because I was sick of paying a few dollars for a magazine with only a page or two of Apple /// information.

To put it quite bluntly, I'm glad you're cancelling those other subscriptions. It's really their own fault for not addressing the needs of the Apple /// community.

Gentlemen,

On 12 April 1983, I purchased a 256K Apple /// system along with an Apple dot matrix printer and other assorted accessories. One of the things which convinced me to buy was your premiere issue of **ON THREE** which happened to be in the showroom. After seeing just that one issue, I felt it was just the type of publication which would help me get the most out of my system and also to get my family involved. Sign me up!

A problem of interest to new buyers: Since I elected to buy the Apple dot matrix printer, I needed a grappler interface to link to the printer. At first, my salesman said I just had to read the manuals to figure it out. As you are probably aware, this was not the case. After spending most of the weekend scouring through manuals, I was told the following Monday that my salesman had it on diskette for me. Well, he came out and installed it for me and everything seemed to be going smoothly. That evening, every boot diskette with the grappler.driver on it locked the system up. We spent two weeks swapping out systems, trying various versions of SOS, placing calls to Charlotte, NC, moving the system from one side of the house to the other, etc. Nothing seemed to work.

By accident last weekend, I happened to turn my printer on the system was locked up, and voila!, it booted. After three weeks the salesman making housecalls and me talking to all kinds of micro we finally figured out. The Apple dot matrix printer has to be on in the select mode in order for the system to boot without intervention. In other words, once you walk into your computer room, turn the printer on and place it in the select mode BEFORE you try booting your Apple /// up. Hopefully, this information will save a of heartache and ill feelings after your initial purchase.

Enclosed will find my check for a subscription to **ON THREE.** Best wishes on new publication.

Sincerely,

Del Brashares
Alabama

Dear Mr. Brashares,

That's the kind of letter I like to hear! I know ON THREE has been having some impact on the sales of the Apple ///, but only letters like yours can assure us that this is really happening.

The note about the Apple Dot Matrix Printer and the Grappler device driver is something that I'm sure many of our readers liked to hear. I'm always amazed by the problems that can happen when interfacing two different pieces of equipment.

Thanks again for your letter.

Good Morning!

Enclosed please find my check covering the next twelve issues of your publication.

After reading issue #2, I suddenly felt that I was no longer stranded on a lonely island.

I have owned my /// since last July, and have not been successful in receiving any communication from Apple or for that matter the dealer who sold me the machine. A search of computer stores and bookstores for material related to my Apple are futile. I was resigned to thinking that we who have the /// are alone and the problems we had were ours to deal with alone.

Thank you for a fine magazine which appears to have the necessary knowledge to solve the problems we have been encountering.

Very Truly Yours,

Michael Gordon
Florida

Dear Mr. Gordon,

With the introduction of ON THREE, Apple /// users will not have to feel alone any more. We're here to help you so if you have any problems in the future please write again.

Dear Mr. Consorti,

You have at least one more supporter in your efforts to get the /// its due place in the microcomputer world. I also get Call-A.P.P.L.E. and Softalk, two fine publications, but their primary focus is on the][in its many forms and I long for some more attention to my machine.

Do you have an effort to build a group library, perhaps one that could be accessed by phone? I can report that "Patching Apple /// Pascal" (J. Jeppson, Softalk, Feb. 1983, p. 190) worked immediately to make my ProFile the Pascal System volume.

In your review of Apple Writer /// you warn that it cannot use more than 128K. This is an important bit of information that I would suggest you include in your "Products Received" feature.

I am enclosing a SASE for your Author Guidelines as well as my check for my subscription.

Sincerely,

Harry T. Hanson
New Jersey

Dear Mr. Hanson,

We are planning a bulletin-board type service in the near future for those of you with modems. Your ideas on this subject will be most welcome.

Dr. Jeppson's Pascal Patch does work very nicely. Just like Quark's Catalyst, it allows you to put all your Pascal files on the ProFile.

Dear Sir,

Congratulations. You have achieved that rare blend of speaking to many varied interests while making the total publication of interest to everyone. I don't think I ever read a magazine from cover to cover until your issue of **ON THREE** arrived.

I am enclosing a check for a subscription, clock and disk of the month. I don't want to miss anything.

Again, thanks for filling a void for those of us with Apple ///'s that have just been waiting for you to come along. I look forward to future issues. I will also start promoting other subscriptions. We can't let this grand start you have made wither and die for lack of support.

Sincerely,

Patrick E. Thomas
Iowa

Dear Mr. Thomas,

Thank you for the letter, we are doing are best to interest everyone. Since word of mouth is the best type of advertising, I very much appreciate your telling other Apple /// owners about ON THREE.

Dear Mr. Consorti,

Enclosed is my $30 check for a subscription to your magazine.

I have been using the PFS series of programs for quite a while and I feel rather qualified, therefore, to be critical of the grade point you gave PFS: Report. In my mind I feel that a B- is too low. In fact, an A would be appropriate. I purchased both an Apple /// and a TRS-80 II to test them side by side for an application in my avacodo grove. The Apple /// won hands down principally because of PFS: Report. The TRS-80 is now sold.

Kindly send me a copy of your "Author Guidelines." I'd like to squeeze some time and write an article on how to make those programs really work.

Let me join your well wishers in congratulating you on a fine publication. There is no question you are filling a need so I hope your subscription file becomes overloaded quickly.

Perhaps you are open to suggestion. Why not have a department for novice owners? I'll bet general interest articles and instruction on programming fundamentals, at a level that the novice can might just be the kind of bait you need to get them hooked. Say it is intermediates but pitch it to the novices. The experts can skip it if want. Mull it over.

Sincerely,

Brooks Lyman
California

Dear Mr. Lyman,

Thank you very much for your recent letter. We are always open to constructive criticism and I welcome your views on the PFS series of software. I look forward to your upcoming article submission.

Your ideas on novices has been duly noted and as early as the June-July issue we will adopt some of these ideas. An entry level column will soon appear to help guide the novices over the sometimes rough waters of learning the Apple ///.

Dear ON THREE,

It was really nice to see the new publication dedicated to the Apple ///. I thought you might like to know that there is a dedicated Apple /// SIG operating online SOURCE at PUBLIC 21 DIRECT. The user group has been operating for over a year. We do charge a membership fee of $20 and the SIG is closed to non-members. I can be contacted through SMAIL (to ST0823) for membership information and sample SIG material.

The SIG maintains an online interactive bulletin board and a twenty plus item data base containing programs and other material for downloading. As an example, the data base contains an auto-logon dialer for use with hard disk, and even a game. The bulletin board provides an online interactive forum for exchange of information, for example a notification of a soon to be released co-processor card (emulates IBM PC), and for online help with Apple /// specific problems. We maintain a listing of over 175 Apple /// native programs (no CP/M programs) currently available. The membership consists of some highly qualified programmers, hardware types, business users, and even an Apple rep. or two.

ON THREE has received very favorable reviews from our members. Keep up the good work!

Sincerely,

Ernest Raba
North Carolina

Dear Mr. Raba,

I'm glad to hear of the Apple /// SIG on the SOURCE. As Apple /// users we need to band together in every way shape and form to help each other.

Dear Mr. Consorti,

Thank you for an excellent publication. I have recently installed your **ON THREE O'Clock** and I am very happy with it, thanks. Another item you might consider offering would be 64K RAM add on chips for those of us with 128K RAM Apple ///'s. There are many 200ns Apple compatible chips that will work with the 5 volt memory board in the ///, all of them much cheaper than the Apple price of $28/chip.

With regard to the question of utilities to edit Business Basic programs that was raised by John Miller in the March issue, most Apple /// owner's have a fantastic editor available in Apple Writer ///. This program can be used to global search/edit/replace as well as insert and move whole blocks of Basic code. In short, all of the features that are available to edit text can be used to edit Basic programs, including all of the Word Processing Language. The trick is to convert the Basic program into a text file that Apple Writer /// read. The following statement inserted as the first statement in the that you want to edit will do the trick:

```
0 CREATE "Program.Name",TEXT:OPEN#1,AS OUTPUT,"Program-
.Name",TEXT:OPEN #1 AS OUTPUT,"Program.Name":OUTPUT#1:
OUTREC=255:LIST 1 TO 63999:CLOSE:END
```

The procedure for doing this is outlined on page 33 of the Basic

manual. The "Program.Name" should be different than the name given to the original Basic program and should include the pathname if you want the text file saved on an alternate disk drive. The reserved word OUTREC sets the record length for the listing to 255 rather than the standard 80 characters that appear on the screen. This keeps the carriage returns from messing up the listing on Apple Writer ///. Once you have entered this line as the first line of your program, RUN it and it will create a text file with the name of "Program.Name". Now all that is left is to boot Apple Writer ///, load your text file, and edit away. When you have finished editing, save your file back to disk. To convert your file back to a Basic program, boot a Basic Disk and just type:

EXEC Program.Name

The disk drive will whirr for a while and each line of text will generate a ')' as it is being read into the Apple. When the disk drive stops, type LIST and your edited creation is now available to be RUN or SAVEd back to disk as a Basic program. Note that if you have any unintelligible lines that Basic does not recognize when you EXEC the text file, you will be unceremoniously given an error and the whole operation will stop. You must then go back to Apple Writer to find your fatal flaw.

There are some interesting things that can be done with WPL on Apple Writer /// (ie. auto line numbering, subroutine library insertion, etc.), but I have found that the most beneficial use of Apple Writer is with very long programs that require major surgery.

Now perhaps someone out there can help me? I am interested in getting FORTRAN running on my ///, but I have not found any way to do it yet. Also I am interested in contacting /// owners who are using the Source. Perhaps we could get together and exchange programs and tips, if so please contact me via SMAIL at CI1759.

Sincerely,

Eric M. Moeller
Montana

Dear Mr. Moeller,

Thank you for your letter, I'm pleased that you like the clock. In respect to the 64K RAM chips, there is a small problem.

The memory boards for the Apple /// are not all the same. The 128K machines that were shipped in the beginning had three rows of memory chips. Later on, these were replaced with a memory board that had ony two rows of chip sockets. If both of these rows were filled you had a 256K machine, otherwise you only had 128K.

Since you can't just plug in chips to the old 128K memory boards to get 256K, there is no way I could offer a memory increase update at this time. With the Apple /// CP/M card you can run FORTRAN on your ///. However, you will have to do quite a bit of patching to get it to work.

There is an Apple /// user group on the Source, it's operating at PUBLIC 21 DIRECT. They do charge a membership fee of $20 and it is closed to non-members. Contact Ernest Raba through SMAIL at ST0823 for membership information and sample material.

Dear Sir,

Enclosed is my check and order for one of your **ON THREE O'Clock's** for my Apple ///.

In addition, I keep hearing rumors that Apple may be discontinuing support of the ///. Do you have any information as to their plans with respect to the equipment and future software. I did receive a letter which indicated a new version of Apple Writer /// with a speller package was coming out in 1983.

Also, I am comsidering upgrading my 128K Apple /// to 256K. Do you have any recommendations as to how I might do this at reduced cost? I would also like to have CP/M capablility.

And, do you have any information on the DataFax file program for the ///? Do you have any recommendations on a good terminal program to use with the Hayes 1200 Smartmodem? Do you know of a Hi-Res Dump program for the Epson MX-100 with Graftrax?

Appreciate your assistance and will look forward to receiving the clock unit.

Sincerely,

Richard N. McKinney
Illinois

Dear Mr. McKinney,

Trust me, Apple is not going to discontinue support of the ///, it is far too fine a machine. I do have some knowledge of future plans for the /// but I'm not at liberty to discuss them. However, what I know has convinced me that Apple is not going to drop the ///.

The new version of Apple Writer /// should be available within a short time. Likewise, Apple Speller /// will be available very soon. At this time the only way to upgrade your machine from 128K to 256K is the Apple way - high price!

I also wish that Apple would lower the cost of the upgrade. The actual cost of the chips and memory board is no doubt much less than $250, so I can't see the reason for such a high margin on this item. You should be able to get an Apple /// CP/M card from any Apple dealer. Currently, this is the only Z-80 card that works with the ///.

We don't have access to the DataFax file program for the ///, so I can't give a recommendation on it quite yet. On the subject of a terminal program for the /// and a Hayes 1200 Smartmodem, Apple has a number of good communications packages for the ///. The one you should buy is contingent on what you are going to use it for. Your best bet would be to ask your local Apple dealer. They should know the one to suit your needs.

Alpine Computing, Inc. sells a Hi-Res dump program for the and the Epson variety of printers. We are going to do a review it next time around. Check the 'Products Received' section of April-May issue for Alpine's address and phone number.

Dear Sir,

I have recently purchased an Apple /// and an Epson MX-100 printer. The salesman who sold me these, convinced me that a interface card would be sufficient. Since that time, I have read articles, including the Epson owners manual, which suggests that we should hook up the Apple /// with a UPIC. Would you explain the difference between the serial and UPIC? What limitations will I face with the serial card?

I am happy there is finally a magazine for Apple /// owners. My

wife and I are looking forward to your next issue of **ON THREE.**

Sincerely,

Thomas Spreitler
Illinois

Dear Mr. Spreitler,

First of all, I hope that you are talking about the built-in serial port that the Apple /// has and not a plug in card because a serial interface to an Epson is fine, I used the built-in serial port to connect to my MX-100 for over two years without any problems.

If you are talking about a plug in serial card that your dealer sold you, I'd bring the darn thing back to the dealer and demand a refund, because he is cheating you! If your Epson is set up to receive a serial signal, it will work fine hooked up to the built-in serial port and does not need another serial or parallel card.

The only reason to buy and use any plug in interface card is to connect a piece of hardware to your computer. Since the MX-100 works fine hooked up to the built-in serial port, there is no need for a plug in card except in the case that you are already using your serial port to connect another item such as a modem to your ///.

If you have any further questions, please don't hesitate in writing again.

Gentlemen,

I am enclosing my check for $30 to cover a one year subscription to **ON THREE.** For some reason, I had not heard that this magazine was about to be published and I just happened to run into the first copy at my local computer store.

While I am in the process of enrolling, may I also send along two items that you may be able to help me with.

My Apple /// is connected to a NEC 7720 Spinwriter. After considerable experimentation I found that it would work fine if the NEC cable was connected to the modem eliminator cable that came with the Apple. However, the '.PRINTER' driver as originally set up by Apple could not be used. Instead a more complete description of the driver configuration had to be supplied by using the '.RS232' driver and alternating the parameters. this however is not my problem - as I said, this combination is now working fine.

In Apple][emulation this same set up gave continual data error signals which I finally ascribed to the fact the "driver configuartion" in the emulation mode (which I could not access) did not provide proper "handshaking" with the printer, so it overflowed the printer buffer with data and stopped it. By choosing 300 baud when the emulation disk is booted, and setting the printer for 300 baud, all went well, but relatively slowly.

Can you offer a solution that will permit me to operate my printer at 1200 baud when I am using the Apple][emulation mode?

The second item concerns my displeasure with a software package - Apple /// Business Graphics sold by Business & Professional Software Company of Cambridge, Mass. I had a hard time getting this package to print anything but when I finally figured out the correct cabling, it transmitted text and numbers okay. I got no help in working this out from Business & Professional Software.

I am still unable to print out graphs and the attitude at Business & Professiona Software is that they have no time to get into my problem, and besides, they don't have an Apple /// with which to experiment! When called upon to produce a graph, several pages of paper are fed through the printer before activity starts. Then the graph that is produced is misshaped. That is, the axes are not uniform and the tics are not evenly spaced. The points appear on the paper but a line that is supposed to connect these points will miss them.

I have tried everything that Business & Professional Software has suggested to no avail and I just thought that you and your readers would like to know about this problem with Apple /// Business Graphics. thank you for any help that you may be able to give me.

Sincerely,

John Lomartire
Connecticut

Dear Mr. Lomartire

Unfortunately I can't give any real solution to your interfacing problem in emulation mode, but I will put the question to our readers. Hopefully someone will respond with an answer. You might try pressuring your local Apple reps. to see if Apple is going to release a new emulation diskette that could fix your problem.

On the subject of Apple /// Business Graphics, I have had similar complaints against this company and product. My only advice here is to take the thing back to your dealer and get him to make it work with your system. If it won't work for him, demand a refund.

It's crazy to pay for something that doesn't work and I'm sure your dealer will do everything in his power to rectify the situation.

Dear Mr. Consorti,

First let me say that I am delighted to have **ON THREE** available. It was and is much needed. You can count on my full support. I plan to call attention to it whenever and wherever I can. Although I certainly would prefer a monthly magazine, I would also prefer a bi-monthly one as against none at all if there has to be a choice.

One suggestion and request. It would be nice if **ON THREE** would let us know what is going on at Apple that is relevant to the ///. For instance, it was through pure luck I found out several months ago that the Utilities Diskette has been updated. And only tenacity and perseverance at continually inquiring led me to find out a few weeks ago that the long awaited SOS Reference Manual and Device Driver Writer's Guide were available. Neither Apple nor their distributors have been of much help in this regard. One exception, it was through Apple I found out about **ON THREE.**

A question. There is an Applegraphics II software which allows, in particular, one to do interactive 3-D graphics. As I understand it, it is in P-Code. I also understand that much Apple][Pascal software can be used on the Apple ///. Would anyone know for certain whether or not Applegraphics II can be used on the Apple /// with Pascal?

Re: things I'd like to see in **ON THREE.** A discussion of the emulation software, particularly in relation to possible modifications. For instance, could the 48K limitation be removed? There is plenty more memory available, could a language card emulation be written? If not, I would like to understand why. Something else I would be interested in reading about is a hardware modification

which would allow me to use a cassette with my Apple /// or with the Apple][emulation mode.

Sincerely,

Andre M. Weitzenhoffer
Oklahoma

Dear Mr. Weitzenhoffer,

Thank you very much for your support, we need all the coverage we can get, and the best kind is a satisfied customer! We will go to a monthly format as soon as possible, hopefully by August.

In the future we will be telling you of relevant happenings in the /// world (updates, new products, etc.). For now let's just say that even with my connections it's hard to find these things out. Thank you for telling me of the Utilities diskette update, I haven't heard about it.

The new SOS manuals are great, if you were using the old photocopied manuals, your programming life will be much easier now! They are very nicely typeset with a lot of great information.

Since the /// has the Apple][Turtle-Graphics Library UNIT, you could use Applegraphics][on the /// if you had access to the source code and could then re-compile it. Even if you could get the source from Apple (you can't) you would be under some serious restraints. Among them, only the Black/White 280 by 192 graphics mode is available. The routines are also much slower than the Apple][counterparts. You can't use it in emulation mode because it needs a 64K system to operate.

We are planning a series on the emulation mode and modifications to the emulation disk. I haven't studied it but I don't believe that a language card emulation could be written due to the format of the ///'s memory. Once we publish this letter maybe someone could also respond about the lack of a cassette port in the emulation mode.

If you have any further questions, please do write again.

Gentlemen,

THANK GOD! Something for the Apple ///! Please send it ASAP.

Thank You,

J.B. Moore
New York

I couldn't resist putting that one in!

Dear Mr. Consorti,

Enclosed is my check to purchase a subscription to **ON THREE**. I found a copy of the January issue and am very impressed.

I encourage you to maintain the advertising policy indicated in your open letter to the readers. This is one way a non-computer expert will have to evaluate products offered by various manufactures. At least one will know if they are advertised in **ON THREE** a knowledgable computer person has reviewed the product and found that it does not contain material flaws.

Business applications and accounting packages would be of greatest interest to the undersigned.

Sincerely Yours,

Charles F. Schreiber, Jr.
California

Dear Mr. Schreiber,

Thank you for your letter, I appreciate your comments regarding our advertising policy. We will have reviews on some major accounting packages in the very near future.

Dear ON THREE,

If my information is correct, there are approximately 80,000 Apple /// users throughout the country. Now, that's a user's group of potentially immense power and it behooves Apple Computer to inform that community of the existence of **ON THREE.** With the recent drop in price of the 256K Apple /// and the burgeoning of significant software, the number of ///ers should grow well beyond the 100,000 mark, thus making **ON THREE's** challenge clear: to produce the best and the most informative computer-related publication around. You've made a good first step in that direction.

Sincerely,

Michael Sexson
Montana

Dear Mr. Sexson,

At this point in time there are approximately 100,000 Apple /// users out there. You're right in your assumption that there is a potential for a very powerful user's group. With help from people like yourself, Apple /// users will soon band together for their common interest.

Sometime in June of this year, Apple will do an Apple /// user mailing that will include a notice about ON THREE. The problem is this: Out of 100,000 Apple /// users out there, less than 20,000 have sent in their registration cards.

Since those cards are the only way Apple knows where Apple /// users are, the mailing will reach less than a fifth of the Apple /// populace. Therefore, if you know of other Apple /// owners, tell them to mail in their registration cards. It's probably the only way they will ever find out about future Apple /// products information.

Dear Bob,

As a businessman I look on my /// as a tool that either works for me, or is ineffective when used for the tasks for which it was purchased. After a short, initial 'Software Shufle' my /// has performed admirably. I am most satisfied. While I am almost entirely a user of 'Pre-Canned' software, your publication will fill several needs that have not been met by anyone else to date.

The first and most important of these needs is one that you addressed in your editorial in the Feb.-March issue. That is, primarily, to bring pressure to bear on Apple Corp. to release all technical info. on the ///, to help those of us who have spent sizeable

dollars on their product to use it more effectively. Apple has not exactly bent over backwards to assist either myself, or my local dealer in answering several question/problems that have arisen since my purchase.

The non-existant SOS manuals have been a major irritant to me, as has been the non-existant Backup ///, and the non-existant Clock/Calendar. Now that Backup /// is online, and I am buying your clock the minor irritants are over. The major one remins, however. Apple Computer Corp. still is considerably less supportive than the good software houses (State-Of-The-Art, and Quark come immediately to mind), and yet I spent a lot more money with Apple than I did on software. Have you tried calling tech. support with a question? "This is technical support, all of our technicians are busy right now. Please leave your name and question, and if we get any free time in the next decade, we will return your call . . ."

The second need that you will/are filling is the need for a forum of ideas. At three dollars per issue, all I have to do is find one little tidbit like Apple Writer ///'s I/O error as the result of writing too many times to the disk, and your magazine has paid for itself. All the other stuff is gravy!

Finally, just by existing you are telling Apple Corp. that we Apple /// users give a damn what happens to the ///, and are clamoring for support. Thanks to you we now have our own 'Micro-Lobby' in the hallowed halls of Cupertino.

Keep up the good work, I look forward to my subscription. Now if you could just do something about that SOS manual.

Sincerely,

Ted Simpson
North Carolina

Dear Mr. Simpson,

For what it's worth, I think the only thing that can be said about Apple's support of the /// is the following line. Apple has grown so big, so quick that it is having problems keeping up with the demand for help in these matters.

Much of it isn't Apple's fault, their rep. firms around the country and even the local dealers haven't supported the /// as they should. I've talked with the people at Apple and they are aware of the problem. It's just going to take some time before it's corrected.

The SOS manuals are available now. You will probably have to order them from your dealer though. The product number is A3L0027. After working with them for a couple of months I don't think they are for the average user. They contain much of the technical details of SOS and are therfore hard reading for most people. ///

Products Received: Continued . . .

STOCK PORTFOLIO SYSTEM

The STOCK PORTFOLIO SYSTEM by Smith Micro Software is a personal investment accounting, record keeping and control system designed to provide you with the facts you need to make informed investment decisions. It covers a wide range of investments, including stocks, bonds, options, multiple CD, bank, credit union and money market accounts.

It is menu driven and very flexible. You can let the STOCK PORTFOLIO SYSTEM access the Dow Jones News/Retrieval service to get stock quotes. The terminal mode allows you to reach out beyond your Apple for direct access to the nation's financial news and other services.

The STOCK PORTFOLIO SYSTEM has a suggested retail price of $185 and is available for the Apple II, Apple /// and IBM PC.

Smith Micro Software, P.O. Box 604, Sunset Beach, California 90742. (213) 592-1032.

VersaForm

VersaForm is the Business Form Processor. Utilizing the familiarity and structure of existing paper forms, VersaForm accelerates the speed and accuracy of processing information. Because it emulates current paper handling procedures, VersaForm is very easy to tailor to specific uses.

This package makes it possible for you to do your business data handling tasks on a computer, using what you already know. Instead of requiring that you enter the world of the programmer, VersaForm works in the familiar world of business forms and deals with your data just the way you do now - on paper, but with the speed and power of a computer.

A complicated system, the VersaForm package takes a while to get to know, but it is worth it. To be reviewed in the June-July issue.

Applied Software Technology, 14125 Capri Drive, Suite 4, Los Gatos, California 95030. (408) 370-2662. ///

REVIEW ON: Everything: Continued . . .

Equipment used in this review:

128K Apple ///
1 external floppy drive
1 ProFile hard disk

Program: CRITICAL PATH SCHEDULING for the Apple ///
Version: Revision #4, 4/6/82, Release 2.0
Contents: Boot and program diskettes, User's manual.
Programming language: Business Basic
Operating System: Standard SOS
Copy Protected: No
Disk Warranty: 90 days
Backup disk included: No
Cost: $495

The Bottom Line

CRITICAL PATH SCHEDULING

Performance: Good
Documentation: Good
Ease of use: Good
Error Handling: Good
Over All Rating: B

The Editor's Block: Continued . . .

One last item before I go. We will publish notices of any Apple /// club or user group, so if you know of one - please tell us! The more local help an Apple /// user can get the better.

That's all for now. Remember, if you have any questions or problems that your dealer can't answer - Ask ON THREE! ///

DOS File List

by Bob Consorti

Many Apple /// utility programs allow the user to access in one way or another Apple][DOS files. But none of them (to my knowledge) let the user see what files are on the DOS diskette. To the average Apple /// owner this means that they must do one of two things, 1 - Remember the exact file name, or 2 - Use the emualtion disk to boot the DOS disk and then do a catalog!

Since Apple][file names can be longer (30 vs. 15) than their Apple /// counterparts, the chance of getting it right is **slim.** Also, booting the emulation disk takes time and is a big hassle. What's a poor Apple /// user to do?

The answer is quite simple - read on! Since the utility programs mentioned above usually are written in Pascal I wrote a routine in that language that allows you to list the files on a DOS 3.3 diskette. Just as in the last issue we will implement it as a UNIT so that all your Pascal programs can use it.

Listing #2 is the actual Pascal Intrinsic UNIT that does the work. Now all programs written in Pascal can give the user the option of seeing the catalog of an Apple][DOS diskette. Like the SOS file lister from the last issue, it will send the output wherever you want ('.CONSOLE','.PRINTER', disk-file . . .) and list the files on an arbitrary sized viewport.

Program Listing #1 is an example of a Pascal program that uses the unit to list Apple][DOS files. If you have been following along, it is very similar to the SOS file listing test program from the last issue. Very simply, it just calls the routine 'List_DOS_Directory' with the appropriate parameters and the unit returns after listing the DOS directory (if possible).

The test program and the DOS file listing UNIT use many of the same routines of the SOS file lister from the last issue, so I will not go into details of how the program works. The Pascal structure of DOS file entries was taken from the November/December 1981 issue of Call-A.P.P.L.E. Dr. Wo wrote the original Apple][implementation of the DOS file list, and I did all the Apple /// changes.

If you want to use this program and UNIT with your other Pascal programs, first type in Program **Listing #2** and compile it with the name 'DOS.STUFF'. Next type in Program **Listing #1** and compile it with the name 'TEST'. Before you can use the program, you must first add the UNIT to your SYSTEM.LIBRARY.

Going through it step by step, at the main command level X)ecute the file 'LIBRARY.CODE'. When the screen prompts you with 'Output file -→', enter 'NEW.LIB'. Make sure that the diskette you are putting 'NEW.LIB' on has enough room to hold the sum of the old library and a little bit more. When the screen prompts you with 'Input file -→', enter '.D1/SYSTEM.LIBRARY'. Now type '=' to copy all the library segments from the old library to the new one.

When the disk drives have stopped making noise, type 'N' to choose a new file and then 'DOS.STUFF' and now press 'RETURN'. Next look on the screen under the prompt 'Output file - ' and find the first two slot numbers that aren't occupied. Type 1 <space> 1st empty slot number <space> and then 2 <space> 2nd empty slot number <space>. Finally type 'Q' to quit and then enter any copyright notice you want to include. Lastly, use the filer to remove the old 'SYSTEM.LIBRARY' and transfer this new library onto your system diskette with the new name 'SYSTEM.LIBRARY'.

Once you have correctly installed it, all of your programs can use it. Now you can execute Program **Listing #1** and test out the new UNIT by cataloging DOS diskettes. If you don't want to type in the programs, or don't have Pascal you can purchase the April-May Disk Of the Month which has all the necessary programs in it. Once again that's all for now. Next time, who knows? I've been working on assembly language routines that will enable programs to save and load text screens to and from disk to memory. Hopefully it will be completed in time for the next issue. ///

DOS File List: Program Listing #1

```
PROGRAM DOS_List;

{ ************************************************************************** }
{ *                                                    ------------------  * }
{ *  Disk Pak3: List_it - DOS_List_Stuff Test Program | Copyright 1983 by |  * }
{ *  ---------------------------------------------- |  O N   T H R E E  |  * }
{ *  by Bob Consorti                                |  April-May, 1983  |  * }
{ *                                                  ------------------  * }
{ *                                                                      * }
{ *  This program uses the Intrinsic unit 'DOS_List_Stuff' to list the   * }
{ *  contents of any DOS directory.  Note that you can define the number * }
{ *  of lines to be listed per page.  Thus, you can set a viewport and   * }
{ *  list the files according to the size of that text window.           * }
{ *                                                                      * }
{ ************************************************************************** }

USES DOS_List_Stuff;          { Contains the routines to list a DOS directory }
                              { You must install it into your SYSTEM.LIBRARY. }
                              { See article for instructions. }
CONST Bell = 7;               { Causes a beep on the internal speaker }
      Top_viewport = 2;       { Sets the top of the currently defined viewport }
      normal = 17;            { Sets normal  video output (White on Black) }
      inverse = 18;           { Sets inverse video output (Black on White) }
      Escape = 27;            { The ASCII number of the ESCAPE character }
      Clear_viewport = 28;    { Homes the cursor and clears the viewport }

TYPE Counter = INTEGER;

VAR Out_Path, Num_Str: STRING;                          { The output files }
    In_Num, Error_code: INTEGER;        { returned by the Unit DOS_List_Stuff }
    Lines_on_window: INTEGER;    { The number of lines in the current viewport }


PROCEDURE Set_titles;     { Sets the main page heading for the entire program }
VAR i: Counter;

BEGIN
  WRITE (CHR (Clear_viewport));
  WRITE ('Disk Utility Pak3');
  GOTOXY (66, 0);
  WRITELN ('Copyright 1983');
  WRITE ('by Robert Consorti');
  GOTOXY (68, 2);
  WRITELN ('by ON THREE');
  FOR i := 1 TO 10 DO
    WRITE ('--------');
  WRITE (CHR (Top_viewport))
END; { of PROCEDURE Set_titles }


PROCEDURE Print_Error;  { Routine to print out the error message }
VAR Ch: CHAR;

BEGIN
  WRITELN (CHR (Bell));
  WRITELN ('WARNING: Error #', Error_code);
  GOTOXY (0,23);
  WRITE (CHR (inverse), 'Press any key to continue', CHR (normal));
  READ (KEYBOARD, Ch)
END; { of PROCEDURE Print_Error }


FUNCTION Str_to_int (num_str: STRING): INTEGER;

VAR Place_num, Temp_int: INTEGER;
    i, Pos_cnt: counter;

BEGIN
  Temp_int:= 0; Pos_cnt := 0;
  FOR I := LENGTH (Num_str) DOWNTO 1 DO
    BEGIN
      Place_num := ORD (Num_str [i]) - ORD ('0');
      Pos_cnt := Pos_cnt + 1;
      Temp_int := Temp_int + TRUNC (PWROFTEN (Pos_cnt - 1)) * Place_num
    END;
  Str_to_int := Temp_int
END; { of FUNCTION Str_to_int  }


PROCEDURE Get_Paths;
BEGIN
  WRITELN (CHR (Clear_viewport));
  WRITELN ('(RETURN for ''.CONSOLE'', ESCAPE RETURN to exit)');
  WRITE ('Enter where I should send the listing -> ');
  READLN (Out_Path);
```

```
  WRITELN;
  IF (LENGTH (Out_Path) = 0) THEN
    Out_Path := '.CONSOLE';
  IF (Out_Path [1] = CHR (Escape)) THEN
    EXIT (PROGRAM);
  WRITELN ('Enter the unit number of the disk to list ');
  WRITE ('Built in disk is 4, 2nd drive is 5, etc. --> ');
  READLN (Num_Str);
  In_Num := Str_to_int (Num_Str)
END; { of PROCEDURE Get_Paths }

BEGIN { Main program }
  Set_titles;
  Lines_on_window := 20;   { There are twenty lines on this viewport }
  REPEAT
    Get_Paths;
    Error_code := 0;
    List_DOS_Directory (In_Num, Out_Path, Lines_on_window, Error_code);
    IF (Error_code <> 0) THEN
      Print_Error
  UNTIL (2 + 2 <> 4)
END. { Of PROGRAM DOS_List }
```

DOS File List: Program Listing #2

```
UNIT DOS_List_Stuff;

{ ************************************************************************** }
{ *                                                  ------------------    * }
{ *  Disk Pak3: DOS File List                      ¦ Copyright 1983 by ¦   * }
{ *  ------------------------                      ¦  O N   T H R E E  ¦   * }
{ *  by Bob Consorti                               ¦  April-May, 1983  ¦   * }
{ *                                                 ------------------     * }
{ *  This Intrinsic Unit gives any Pascal program the ability to list the  * }
{ *  files on any Apple ][ DOS diskette.                                   * }
{ *                                                                        * }
{ *  Original Pascal definition for the DOS file structure is from the     * }
{ *  the November/December 1981 edition of Call-A.P.P.L.E. by Dr. Wo.  All * }
{ *  Apple /// changes and enhancements by Bob Consorti.                   * }
{ *                                                                        * }
{ *  Please read the article and the program List_It to see how to install * }
{ *  and use this Intrinsic Unit.                                          * }
{ *                                                                        * }
{ ************************************************************************** }

{$E+ }        { This compiler option allows for private files within the UNIT }

INTRINSIC CODE 25; { Only one segment; files are private to this UNIT }
                   { because of the 'E+' compiler option shown above. }

INTERFACE

PROCEDURE List_DOS_Directory (VAR Unit_Num: INTEGER; VAR Out_Path: STRING;
                              VAR Lines_on_window, Error: INTEGER);

IMPLEMENTATION

{ ************************************************************************** }
{ *                                                                        * }
{ *  The procedure 'List_DOS_Directory' is PUBLIC and can be used by your  * }
{ *  Pascal host program as follows:                                       * }
{ *                                                                        * }
{ *  INPUT to List_DOS_Directory:                                          * }
{ *    1) Unit_Num - The Unit Number of the disk drive with the DOS disk.  * }
{ *    2) Out_Path - Where to send the listing.                            * }
{ *    3) Lines_on_window  - The current number of vertical lines in the   * }
{ *                          viewport.  Used to determine where to make a  * }
{ *                          page break when listing to the '.CONSOLE'.    * }
{ *                                                                        * }
{ * Output from List_DOS_Directory:                                        * }
{ *   1) The listed directory.                                             * }
{ *   2) Error - The error code (as indicated by IORESULT) for the last    * }
{ *              completed Input/Output operation.                         * }
{ *                                                                        * }
{ ************************************************************************** }

PROCEDURE List_DOS_Directory;
CONST normal = 17;          { Sets normal  video output (White on Black) }
      inverse = 18;         { Sets inverse video output (Black on White) }
      esc = 27;             { Ascii value for the key 'ESCAPE' }
      clear_viewport = 28;  { Homes the cursor and clears the viewport   }
      Dos_max = 30;         { Maximum length of a DOS file name }

TYPE
  Byte      = 0..255;
  Dirrange  = 0..105;  { Range of entries in a DOS directory }

  Sectbuffer = PACKED ARRAY [Byte] OF Byte;
  Blockbuffer= PACKED ARRAY [1..512] OF Byte;

  Link = PACKED RECORD { Used to designate track/sector combinations }
           Tracknum: Byte;
           Sectnum:  Byte
         END;

  Dosfilekinds = ( DOS file types )
             (Volinfo, Unknown, Dftext, Dfinteger, Applesoft, Binary);

{ Pascal format for the information contained in a DOS directory entry }

  DosDirentry = PACKED RECORD
    CASE Dfkind: Dosfilekinds OF
      Volinfo: ( This is the volume info )
        (Dunitnum: Byte; Dnumentries: Dirrange);
      Unknown, Dftext, Dfinteger, Applesoft,
      Binary:
        (File_Ts]: Link;  { Location of the files Track-Sector list }
         Locked: Boolean; { Designates whether the file is locked }
         Name: STRING [Dos_max];
         Sectorcount: Byte]         { Number of sectors allocated }
  END;
  Dosdirectory = ARRAY [Dirrange] OF DosDirentry;

VAR
  Device:  TEXT;          { Where to send the listing }
  Dosdir:  Dosdirectory; { The current DOS directiry }
  Ioerror, Line_count: INTEGER;


  PROCEDURE Trap_IO_error;    { If an error occurs, leave the unit }
  BEGIN                       { with an appropriate error number.  }
    IF (IORESULT <> 0) THEN
      BEGIN
        Error := IORESULT;
        CLOSE (Device);
        EXIT (List_DOS_Directory)
      END
  END; { Of PROCEDURE Trap_IO_error }


  PROCEDURE Set_Out_device;    { Sets the appropriate output file }
  VAR i: Byte;

  BEGIN
    IF (LENGTH (Out_Path) = 0) THEN
      BEGIN
        Error := 7; { An illegal Pathname }
        EXIT (List_DOS_Directory)
      END
    ELSE
      BEGIN
        FOR i := 1 TO LENGTH (Out_Path) DO
          IF (Out_Path [i] IN ['a'..'z']) THEN
            Out_Path [i] := CHR (ORD (Out_Path [i]) - 32);
        IF ((Out_Path <> '.CONSOLE') AND (Out_Path <> '#1')) THEN
          IF ((Out_Path <> '.PRINTER') AND (Out_Path <> '.SPRINTER') AND
            (Out_Path <> '.PPRINTER') AND (Out_Path <> '#6')) THEN
              IF (POS ('.TEXT', Out_Path) = 0) THEN
                IF (LENGTH (Out_Path) < 11) THEN
                  Out_Path := CONCAT (Out_Path, '.TEXT');
($IOCHECK- )
        REWRITE (Device, Out_Path);
        CLOSE (Device, LOCK);
        Trap_IO_error;
        REWRITE (Device, Out_path);
($IOCHECK+ )
        Trap_IO_error
      END
  END; { Of PROCEDURE Set_Out_Device }

  PROCEDURE New_Page (Message: STRING);
  VAR Ch: CHAR;
                { Prompts the user to press a key for more files, or to end }
  BEGIN
    GOTOXY (0, 23);
    WRITE (CHR (inverse), Message, CHR (normal));
    READ (KEYBOARD, Ch);
    WRITE (CHR (clear_viewport))
  END; { Of PROCEDURE New_page }

  PROCEDURE Check_console;   { Checks for various options }
  BEGIN
    IF ((Out_Path <> '.CONSOLE') AND (Out_Path <> '#1')) THEN
      WRITE ('.')
    ELSE
      IF (Line_count = Lines_on_window) THEN
        BEGIN
          New_page ('Press any key for more');
          Line_count := 2 { The end of a page - so make a new one. }
        END
  END; { Of PROCEDURE Check_console }


  FUNCTION Readtrksec (Trksec: Link; VAR Sb: Sectbuffer;
                       VAR Ioerror: INTEGER): Boolean;
  { this function reads the sector number 'Trksec.Sectnum' from
    tracknumber 'Trksec.Tracknum' on disk drive number 'Unit_Num' }

  VAR
    Block: Blockbuffer;
    Blocknum, Offset: INTEGER;

  BEGIN
    With Trksec DO
      BEGIN { Compute the half-block corresponding to the desired sector }
        IF (Sectnum IN [0, 15]) THEN
          Blocknum := Sectnum
        ELSE
          Blocknum := 15 - Sectnum;
        IF (ODD (Blocknum)) THEN
          Offset := 256
        ELSE
          Offset := 0;
        { Now compute blocknum offset from track 0 }
        Blocknum := (Blocknum DIV 2) + 8 * Tracknum
  END; { With Trksec DO }
```

Program listing continued on next page.

```
{$I-}
   UNITREAD (Unitnum, Block, SIZEOF (Block), Blocknum);
{$I+}
   Ioerror := Ioresult;
   IF NOT (Ioerror = 0) THEN
     Readtrksec := False
   ELSE
     BEGIN { Write into the sector buffer }
       MOVELEFT (Block [Offset * 1], Sb, SIZEOF (Sectbuffer));
       Readtrksec := True
     END
 END; { Of FUNCTION Readtrksec }


 PROCEDURE Displayentry (De: Dosdirentry);
 BEGIN
   WITH De DO
     BEGIN
       IF Locked THEN
         WRITE (Device, '*')
       ELSE
         WRITE (Device, ' ');
       CASE Dfkind OF
         Dftext:    WRITE (Device, 'Text file');
         Dfinteger: WRITE (Device, 'Integer  ');
         Applesoft: WRITE (Device, 'AppleSoft');
         Binary:    WRITE (Device, 'Binary   ');
         Unknown:   WRITE (Device, 'Unknown  ')
       END;
       WRITE (Device, Sectorcount: 6, ' ');
       WRITELN (Device, Name);
       Line_Count := Line_Count + 1;
       Check_console
     END
 END; { Of PROCEDURE Displayentry }


 PROCEDURE Displayheader;
 BEGIN
   WRITE (Device, ' Type');
   WRITE (Device, ' Sectors': 11);
   WRITELN (Device, ' File name');
   WRITELN (Device, '---------------------------')
 END; { Of PROCEDURE Displayheader }


 PROCEDURE Displaydir;
 VAR
   Cumsectors: INTEGER;
   Count: Dirrange;

 BEGIN
   Cumsectors :=0;
   IF (Dosdir[0].Dnumentries = 0) THEN
     BEGIN
       WRITELN (Device, 'The directory is empty!');
       New_Page ('Press any key to continue')
     END
   ELSE
     BEGIN
       Displayheader;
       FOR Count := 1 TO Dosdir [0].Dnumentries DO
         BEGIN
           Displayentry (Dosdir [Count]);
           Cumsectors := Cumsectors + Dosdir [Count].Sectorcount;
         END;
       WRITELN (Device);
       WRITE (Device, Dosdir [0].Dnumentries,' files on disk, ');
       WRITE (Device, Cumsectors,' sectors in use');
       New_Page ('Press any key to continue')
     END
 END; { Of PROCEDURE Displaydir }


 PROCEDURE Catalog;
 CONST
   Nextlink   = 1;  { Relative byte 1 of directory sector is link to
                      the next directory sectory }
   Zerobase   =11;  { First byte of file info in a directory sector }
   Entrylength=35;  { DOS directory entries occupy 35 bytes }
   Maxindex   = 7;  { Maximum of 7 directory entries in a sector }

   Space      = 32; { ASCII space }
   Tilde      =126; { ASCII Tilde }

 TYPE
   Indexrange = 0..Maxindex;
   Entrybuffer = PACKED ARRAY [1..Entrylength] OF Byte;

 VAR
   Sectorindex: Indexrange;
   Entrybase:   Byte;
   Dir_Link:    Link;
   Dir_Sector:  Sectbuffer;
   Nextentry:   Entrybuffer;
   Entrycount:  Dirrange;


 FUNCTION Eodir (Dirlink: Link): BOOLEAN;
 BEGIN
   With Dirlink DO
     Eodir := ((Sectnum = 0) AND (Tracknum = 0))
 END; { Of FUNCTION Eodir }


 FUNCTION Eodirsector (VAR Index: Indexrange; VAR Dirsector: Sectbuffer;
                       VAR Entrybase: Byte): BOOLEAN;
 VAR
   Nofile: BOOLEAN;

 BEGIN
   Nofile := True;
   WHILE (Nofile AND (Index < Maxindex)) DO
     BEGIN
       Index := Index + 1;
       Entrybase := Zerobase + (Index - 1) * Entrylength;
       Nofile := (Dirsector [Entrybase] IN [0, 255])
     END;
   Eodirsector := Nofile
 END; { Of PROCEDURE Eodirsector }


 PROCEDURE Fill_Dir_Entry (VAR De:Dosdirentry; VAR Eb: Entrybuffer);
 CONST
   Offset = 4; { Relative byte 3 is the beginning of the file name }

 VAR
   J,Kind: Byte;
   Nonblank: 0..Dos_max;

 BEGIN
   WITH De DO
     BEGIN
       Filetsl.Tracknum := Eb [1]; { Here is the track number }
       Filetsl.Sectnum :=  Eb [2]; { Here is the sector number }
       Kind := Eb [3];             { Here is the file type }
       IF NOT ((Kind MOD 128) IN [0, 1, 2, 4]) THEN
         Dfkind := Unknown
       ELSE
         CASE (Kind MOD 128) OF
           0: Dfkind := Dftext;
           1: Dfkind := Dfinteger;
           2: Dfkind := Applesoft;
           4: Dfkind := Binary
         END;
       IF ((Kind DIV 128)=1) THEN
         Locked := True
       ELSE
         Locked := False;
       FOR J := 0 TO (Dos_max - 1) DO
         BEGIN           { Set the high bit low }
           Eb [Offset + J] := Eb [Offset + J] MOD 128;
           { Now, eliminate any wierd characters }
           IF NOT (Eb [Offset + J] IN [Space..Tilde]) THEN
             Eb [Offset + J] := Space
         END;
       { Find the leftmost trailing blank in the name field }
       Nonblank := -SCAN (-Dos_max, <> ' ', Eb [Offset + Dos_max - 1]);
       { Non_blank=0 if and only if there are no trailing blanks }
       { Initialize the length of 'name' }
{$R-}
       Name [0] := CHR (Dos_max - Nonblank);
{$R+}
       { Finally move in the name }
       MOVELEFT (Eb [Offset], Name [1], LENGTH (Name));
       Sectorcount := Eb [34]  { Here is the sector count (MOD 256) }
     END { With De DO }
 END; { Filldirentry }


 BEGIN { Catalog }
   WRITE (CHR (clear_viewport));
   IF ((Out_Path <> '.CONSOLE') AND (Out_Path <> '#1')) THEN
     WRITE ('Writing.');
   WITH Dir_Link DO
     BEGIN
       Tracknum := 17;     { Beginning track of the DOS directory }
       Sectnum :=  15      { First sector of the DOS directory }
     END;
   Entrycount := 0;
   WHILE NOT Eodir (Dir_Link) DO
     BEGIN
       IF NOT Readtrksec (Dir_Link,Dir_Sector,Ioerror) THEN
         Trap_IO_error
       ELSE
         BEGIN
           Sectorindex := 0;
           WHILE NOT Eodirsector (Sectorindex, Dir_Sector, Entrybase) DO
             BEGIN
               MOVELEFT (Dir_Sector [Entrybase], Nextentry, Entrylength);
               Entrycount := Entrycount+1;
               Filldirentry (Dosdir [Entrycount], Nextentry)
             END
         END; { Of ELSE BEGIN }
       WITH Dir_Link DO
         BEGIN
           Tracknum := Dir_Sector [Nextlink];
           Sectnum :=  Dir_Sector [Nextlink+1]
         END
       END;
   WITH Dosdir[0] DO
     BEGIN
       Dnumentries := Entrycount;
       Dunitnum := Unitnum
     END;
   Displaydir
 END; { Of PROCEDURE Catalog }
```

Program listing continued on page 18

Changing The Character Of The Apple Dot Matrix Printer. (and the Prowriter)

by Timothy S. Smith

Introduction:

When I first purchased my Apple /// many moons ago I was very impressed at the ///'s ability to change its video character fonts so easily. Then when I purchased my **Apple Writer ///** program, there they were again, gothic, slant, stop and inverse character fonts, but alas only for my video viewing pleasure. Then recently I purchased the new Apple DMP (Dot Matrix Printer). Hope of gothic characters in my text again appeared on the horizon. But then I began to read the 27 page Apple DMP Owners Manual. The DMP manual is by far one of the most disappointing efforts I have ever seen from Apple. Yet, hope lives on in spite of adversity. While I began to study the small folded DMP reference card, I said to myself "what's this under character commands, Load Custom Character(s)?" Light at last!

With the sliver of support from the DMP reference card I began to experiment with transferring the screen font data to the printer. This proved to be very difficult since not enough information had been provided with the "Operator's Manual"to operate this feature of the Dot Matrix Printer. After some investigation I found that my authorized Apple dealer had the information I so desperately needed, printed in an Apple products manual dated June 24, 1982. Oddly enough, I had found most of the data entry pattern prior to gaining the additional dealer support.

Technical:

If you don't care how it was done and just want to try it, skip ahead to "Operation". Before transferring the video font files from the disk to the DMP you must examine the nature of the data in the file. The video character fonts are made up of a matrix 8 bits high by 7 bits wide. This is illustrated on page 166 of the Standard Device Drivers manual. In the font file for each of the 128 possible character representations are 8 bytes of data, one for each of the 8 rows high. Because the screen is scanned from the top down, so are the font format bytes. The extra bit in each row, the high order byte, determines the inverse display condition.

The normal DMP character font on the other hand is constructed from left to right due to the mechanical action of the printer. Therefore the bytes of data for the DMP are set 1 column of data at a time. The low order byte is the top edge of the character, allowing the high order (bit 7) to determine an underline. Which 8 of the 9 DMP print wires are used is determined by whether or not the character font uses decenders or not.

I chose the Pascal language to construct the conversion program due to it's extremely structured nature. This allows for faster more orginized data transfer.

In converting the video character font for the DMP the first order of business was to create a matrix for the entire character set. An array of 128 character definitions is read into memory. Each definition is in itself 8 x 8 bits. Therefore the array is CH [CR, Row, Column] OF 0..1, where CH is the entire matrix set. CR is the ASCII value of the character defined by Row and Column. Each entry in the array is either a 0 or 1 (cell bit off or on). The font file is read using the low level Blockread function. All of the Bytes are automatically converted into a large single dimension array by the use of the, PACKED ARRAY OF 0..1, variable. Once read into the buffer variable the font cell bits can be transferred to the multi-dimension matrix array CH. Now you have a complete character set in memory in bit image. This bit image could be edited and restored in any order, even re-saved to the same file from which it came.

The reason I have placed all of the font data in memory, as bits in an array, is so that the font cell definitions can now be saved in a different byte structure. Remember the printer must receive its data from the left side, not the top as the console. Transmitting the character font from memory to the printer is now easy. Just play out the bits in a different order creating a new series of 8 bytes for each character. But wait, the video font is only 7 bits wide. The console will automatically provide 1 dot space between characters, but not so with the DMP. The 8th byte is 0 to provide spacing between characters. When moving an inverse character set the 8th bit should be set to 127 or 255 depending on the 8th (bottom) print position condition.

Now before sending the font bytes to the DMP you have to get it's attention, otherwise you get lots of garbage on the paper and lots of paper on the floor as well. The Attention code listed in the DMP reference card is <ESC> I. This is sent by printing an ASCII value for escape (27) followed by the capitol I character (or ASCII 36). The printer must first be opened like an interactive file, RESET (Printer,'.PRINTER'). For a better understanding on printer access from Pascal, review pages 163-164 in the Pascal Programmers Volume #1. After the printer device is open you then Write the attention code to it (e.g. WRITE (Printer, CHR (27), 'I');). If you intend on using the alternate custom font area, ASCII values 160-239, you must also precede the attention code with <ESC> - to inform the printer your characters will be no more than 8 bytes wide (e.g. WRITE (Printer, CHR (27), '-');). If you send a custom character set which will be wider than 8 bytes, up to 16 bytes wide, you must precede it with <ESC> + (e.g. WRITE (Printer, CHR (27), '+');).

After you have the printers attention you can begin sending the <list> as the reference card calls it. This list is the ASCII code definition byte, length byte and then 8 font cell definition bytes (See illustration below). This pattern is repeated until the entire character set has been transfered. To inform the printer that you have completed the task an ASCII code definition byte of ASCII 4 is sent. In the example program the character bytes are transferred using the Unitwrite procedure to avoid character conversions done automatically in Pascal (See page 194 Pascal Programmers Manual #1).

Operation:

If you are an Apple][owner you need a friend with an Apple ///. To use this program you will need an, Apple ///, Pascal, and perhaps Apple Writer ///. You could also try your friendly Apple Dealer. He may have already have read this. Ok Apple /// owners here we go. To begin, the Pascal program in this article must be typed into the Pascal editor and then compiled. No special library functions were used. After the program has compiled successfully, execute the program. Your first choice is to [C]onvert a font file to proper code for the Apple DMP printer or [L]oad a previously converted file from disk.

The conversion operation is simple. You will be asked for the complete path name of a font file (e.g. .D2/ROMAN). If you need further explanation of the path name see your Apple /// Owners Guide or Pascal Programmers Manual. The next question is whether you would like this set converted to [I]nverse or [N]ormal. Inverse prints a black space containing a white letter. By the way selecting Inverse for the Inverse character font is really strange, because of the 8th byte (See technicl notes above). After making your selection the disk drive will whir into action and the display will show (in standard video characters) the font characters that are being converted. When this loading/converting process is done you will be asked to choose whether to send the font data directly to the [P]rinter or the [D]isk for later use. Sending the font to the disk will not load the font into the printer. You will use the [L]oad option to load the font from the disk to the printer.

When saving the font to the disk you will be offered the opportunity to create a text file for conversion to Apple][mode. Converting the Apple /// text file to Apple][text as well as moving it to an Apple][disk is easy, that is if you own a text transfer program like the utility disk which comes with Apple Writer ///. This utility disk when booted will offer you several choices for file conversion #2 of which is Apple /// files to Apple][files. This program works best with multiple disk drives. It's best to use drive 1 for your Apple][Dos 3.3 formatted disk. To convert the font textfile created with the pascal program select #2. Then answer the Apple /// volume question with the proper drive # (and subdirectory path, if any). The program will then ask for filenames. When answering these two questions remember the file naming rules for each type of file. When in doubt use the Apple /// filename for both. Then you will be instructed to place the proper diskettes into their proper disk drives (e.g. Apple][diskette in Drive #1 and Apple /// formatted disk in drive #2). Once moved to an Apple][disk the process of loading the font from the disk to the printer is a simple one. Just enter and run the Applesoft program in this article.

Once the new custom font is loaded into the printer it can be tested with either program. The program will print for you the complete character set followed by a message you can enter. Great uh? Well almost . . . there is one drawback. Only one complete custom character set can be held by the printer at one time. The custom font will remain intact until the printer is turned off or a new custom font downloaded. Run the program listed then try Apple Writer. To turn on the new custom font print <ESC> '. To turn it off print <ESC> $. If you are using Apple Writer you will need to use <CTRL> V to imbed the <ESC> character in your text.

Now Byte, Apple, Roman, Gothic, Slant, Stop and Inverse character fonts can be printed on the Apple Dot Matrix Printer. Now it's time to byte into the long awaited SOS manuals! Documentation at last! 'Happy Bit Byteing' ///

Apple][Special Notes

Note #1: If you are using a Prowriter instead of an Apple DMP and your printer will not accept the custom font, you will need the addition of a 2K x 8 Bit Static Ram (TMM 2016). Early versions of the Prowriter were shipped without this chip, and supporting documentation for the custom font feature. This chip, about the size of a common ROM, is placed in the only empty socket on the main board inside your Prowriter. I suggest that you have your computer dealer handle this process, unless you're very good in small places.

Note #2: The Applesoft program in this article is designed to work with the latest edition of Apple Parallel Interface Cards for the Apple][and Apple ///. Many printer cards installed in Apple][s did not support 8 bit data transfer.

[ILLUSTRATION]

Printer "<list>" Data Format:

Prior to loading Select Font Mode (Max Size).
Send one of the following Codes to the printer:

[ESC] - Selects 1 to 8 bit character width.
Total of 175 possible characters.

(dec) ASCII 32 - 126
ASCII 160 - 239

[ESC] + Selects 1 to 16 bit character width
Total of 95 possible characters.

(dec) ASCII 32 - 126

Character Definition <List> Format:

Precede with [ESC] I

Loop Until Finished Transferring All Characters

```
------>
|          1 Byte  = ASCII Code
|
|          1 Byte  = Length Code
|
|          n Bytes = Char Data (n = Width of Character)
------<-
```

Length Code = Width of Character (1..16) +
32 if Descender Exists.

If Decender Exists printer uses Lower 8 wires instead of upper 8.

After The Last Character Has Been Sent Print ASCII 4 (Chr (4)) To Indicate Completion of Transfer.

CharDownLoad: Program Listing

```
PROGRAM Font_Mover;

{ ************************************************************************ }
{ *                                                  ------------------  * }
{ *  CharDownLoad                                   ¦ Copyright 1983 by ¦ * }
{ *  ------------                                   ¦  O N   T H R E E  ¦ * }
{ *  by Timothy S. Smith                            ¦  April-May, 1983  ¦ * }
{ *                                                  ------------------  * }
{ *                                                                      * }
{ *  This program will download the Apple ///'s screen font to an Apple  * }
{ *  Dot Matrix Printer or Prowriter.  Files may be saved and then later * }
{ *  retrieved for downloading.                                          * }
{ *                                                                      * }
{ *  Read the article for complete information on how to use the program.* }
{ *                                                                      * }
{ ************************************************************************ }

VAR CH: PACKED ARRAY [0..127, 0..7, 0..7] OF 0..1;
    CR, Column, Row,
    Count, Temp     : INTEGER;
    Response        : CHAR;
    Printer         : INTERACTIVE;


PROCEDURE Send;  { Send Font bytes to printer from matrix }
VAR Bits: CHAR;

BEGIN
{$IOCHECK-}
  RESET (Printer, '.PRINTER');
```

```
  WRITE (CHR (28));
  IF IORESULT <> 0 THEN
    BEGIN
      GOTOXY (24, 10);
      WRITELN ('Printer Not Available');
      EXIT (Program)
    End;
  GOTOXY (25, 10);
  WRITELN ('Downloading Font to Printer');
  WRITE (Printer, CHR (27), '-', CHR (27), 'I');
  FOR CR := 32 TO 126 DO
    BEGIN
      GOTOXY (40, 12);
      WRITE (CHR (CR));
      WRITE (Printer, CHR (CR), CHR (8));
      FOR Column := 0 TO 7 DO
        BEGIN
          Count := 1; Temp := 0;
          FOR Row :=0 TO 7 DO
            BEGIN
              Temp := Temp + (CH [CR, Row, Column] * Count);
              Count := Count * 2
            END;
          Bits := CHR (Temp);
          UNITWRITE (6, Bits, 1, 0, 12)
        END;
    END;
  WRITE (Printer, CHR (4));
  CLOSE (Printer)
($IOCHECK+)
END; { of PROCEDURE Send }

PROCEDURE Save;  { Save Font bytes to disk file (In Printer Format) }
VAR FileName: STRING;
    FontFile: PACKED FILE OF 0..255;
    Two     :TEXT;
    Temp, Count: INTEGER;
    Bits    :CHAR;
    AppleII :BOOLEAN;

BEGIN
  WRITE (CHR (28));
  GOTOXY (0, 10);
  WRITE ('Enter File Pathname For Saving :');
  READLN (FileName);
  GOTOXY (0, 10);
  WRITE (CHR (30), 'Save in Apple ][ Text Format (Y/N) ? ');
  REPEAT
    READ (KEYBOARD, Bits);
  UNTIL (Bits = 'y') OR (Bits = 'Y') OR (Bits = 'n') or (Bits = 'N');
  IF (Bits = 'y') OR (Bits = 'Y')
    THEN AppleII := True
  ELSE
    AppleII := False;
  WRITE (CHR (28));
($IOCHECK-)
  IF AppleII THEN
    BEGIN
      REWRITE (Two, FileName);
      IF IORESULT <> 0 THEN
        BEGIN
          GOTOXY (24, 10);
          WRITELN ('Unable To Open File');
          EXIT (Program)
        END;
    END
  ELSE
    BEGIN
      REWRITE (FontFile, FileName);
      IF IORESULT <> 0 THEN
        BEGIN
          GOTOXY (24, 10);
          WRITELN ('Unable To Open File');
          EXIT (Program)
        END
    END;
  WRITE (CHR (28));
  GOTOXY (35, 10);
  WRITELN ('Saving Font');
  IF AppleII THEN
    BEGIN
      GOTOXY (28, 11);
      WRITELN ('In Apple // Text Format')
    END;
  FOR CR := 32 TO 127 DO
  BEGIN
    GOTOXY (40, 12);
    WRITE (CHR (CR));
    FOR Column := 0 TO 7 DO
      BEGIN
        Count := 1;
        Temp := 0;
        FOR Row := 0 TO 7 DO
          BEGIN
            Temp := Temp + (CH [CR, Row, Column] * Count);
            Count := Count * 2
          END;

      { If Saving for Transfer to Apple ][ Save in Text }
                      { Not Byte Format }

        IF AppleII THEN
          BEGIN
            STR (Temp, FileName);
            WRITELN (Two, FileName)
          END
        ELSE
          BEGIN
            FontFile^ := Temp;
            PUT (FontFile)
          END
      END
  END;
  CLOSE (Two, Unprotect);
  CLOSE (FontFile, Unprotect);
($IOCHECK+)
  WRITE (CHR (28))
END; { of PROCEDURE Save }

PROCEDURE Load;  { Load Previously Converted Font file   }
                 { from disk and send it to the printer. }

Var FileName :STRING;
    FontFile :PACKED FILE OF 0..255;
    CH       :CHAR;
    Count, Temp :INTEGER;

BEGIN
  WRITE (CHR (28));
  GOTOXY (0, 10);
  WRITE ('Enter Printer Character File Path :');
  READLN (FileName);
  WRITE (CHR (28));
  GOTOXY (30, 10);

{ Open File And Printer Device }

($IOCHECK-)
  RESET (FontFile, FileName);
  IF IORESULT <> 0 THEN
    BEGIN
      WRITELN ('File Not Found');
      EXIT (Program)
    END;
  RESET (Printer, '.PRINTER');
  IF IORESULT <> 0 THEN
    BEGIN
      WRITELN ('Printer Not Available');
      EXIT (Program)
    END;

{ Move Font From Disk }

  GOTOXY (29, 10);
  WRITELN ('Moving Character Font');
  WRITE (Printer, CHR (27), '-', CHR (27), 'I');
  FOR CR :=32 TO 127 DO
    BEGIN
      GOTOXY (40, 12);
      WRITE (CHR (CR));
      WRITE (Printer, CHR (CR), CHR (8));
      FOR Count:=0 TO 7 DO
        BEGIN
          CH:=CHR (FontFile^);
          UNITWRITE (6, CH, 1, 0, 12);
          GET (FontFile)
        END
    END;
  WRITE (Printer, CHR (4));
  CLOSE (Printer);
  CLOSE (FontFile)
($IOCheck+)
END;  { of PROCEDURE Load }

PROCEDURE Test;  { Test Custom Character Font }

VAR Count: INTEGER;
    Message: STRING;

BEGIN
  WRITE (CHR (28));
  GOTOXY (0, 10);
  WRITE ('Enter Test Message Line :');
  READLN (Message);
  WRITE (CHR (28));
  GOTOXY (36, 10);
  WRITELN ('Testing');
  RESET (Printer, '.Printer');
  IF IORESULT <> 0 THEN
    BEGIN
      WRITE (CHR (28));
      GOTOXY (20, 10);
      WRITELN ('Printer Not Available');
      EXIT (Program)
    END;
  WRITELN (Printer, CHR (27), CHR (39));
  FOR Count:=32 TO 126 DO
    BEGIN
      WRITE (Printer, CHR (Count));
      IF Count = 79 THEN
        WRITELN (Printer)
    END;
  WRITELN (Printer);
  WRITELN (Printer, Message);
  WRITELN (Printer, CHR (27), CHR (36));
  CLOSE (Printer)
END; { of PROCEDURE Test }
```

Program listing continued on next page.

```
PROCEDURE Read_Font; { Read Apple /// Fontfile From Disk And }
                     { Convert TO Bit Matrix Table CH.       }
VAR FontFile: FILE;
                     { Untyped File FOR Blockread Function }
    FilePath: STRING;
                     { SOS File Path - Catalog Path & File Name }
    Inverse: BOOLEAN;
                     { Flag TO Set Character Font TO Inverse Mode }
    Buf: PACKED ARRAY [0..4095] OF 0..1;
                     { Input Buffer As Array of Bits }
BEGIN
  WRITE (CHR (28));
  GOTOXY (0, 10);
  WRITE ('Enter Font File Pathname :');
  READLN (FilePath);
  GOTOXY (0,10);
  WRITE (CHR (30),'Enter As [I]nverse or [N]ormal ?');
  REPEAT
    READ (KEYBOARD, Response);
  UNTIL (Response = 'I') OR (Response ='i') OR
        (Response = 'n') OR (Response = 'N');
  IF (Response = 'I') OR (Response = 'i') THEN
    Inverse := True
  ELSE
    Inverse := False;
  WRITE (CHR (28));
  GOTOXY (32, 10);
  WRITELN ('Opening Font File');

{   Open Fontfile On Disk }

{$IOCHECK-}
  RESET (FontFile, FilePath);
  IF IORESULT <> 0 THEN
    BEGIN
      GOTOXY (28, 10);
      WRITELN (CHR (30), 'Unable TO Open Font File');
      EXIT (Program)
    END;
{$IOCHECK+}

{ Read and Load File TO Matrix }

  WRITE (CHR (28));
  GOTOXY (29, 10);
  WRITELN ('Converting Font Table');
  Teap := BLOCKREAD (FontFile, Buf, 1);
  FOR CR:=0 TO 63 DO
    BEGIN
      GOTOXY (40, 12);
      IF CR > 31 THEN
        WRITE (CHR (CR))
      ELSE
        IF (CR MOD 2) = 0 THEN
          WRITE ('*')
        ELSE
          WRITE (' ');
      FOR Row := 0 TO 7 DO
        FOR Column:=0 TO 7 DO
          Ch [CR, Row, Column] := Buf [CR * 64 + Row * 8 + Column]
    END;
Teap := BLOCKREAD (FontFile, Buf, 1);
FOR CR:=64 TO 127 DO
  BEGIN
    GOTOXY (40, 12);
    WRITE (CHR (CR));
    FOR Row:=0 TO 7 DO
      FOR Column:=0 TO 7 DO
        Ch [CR, Row, Column] := Buf [(CR - 64) * 64 + Row * 8 + Column]
    END;
CLOSE (FontFile);

{ IF Inverse THEN Reverse Matrix }

IF Inverse THEN
  BEGIN
    WRITE (CHR (28));
    GOTOXY (31, 10);
    WRITELN ('Changing TO Inverse');
    FOR CR:=32 TO 127 DO
      BEGIN
        GOTOXY (40, 12);
        IF (CR Mod 2 = 0) AND (CR < 32) THEN
          WRITE ('*')
        ELSE
          WRITE (' ');
        IF CR > 31 THEN
          WRITE (CHR (CR));
        FOR Row:=0 TO 7 DO
          FOR Column:=0 TO 7 DO
            IF CH [CR, Row, Column] =1 THEN
              CH [CR, Row, Column] := 0
            ELSE
              CH [CR, Row, Column]:=1
      END
  END
END; { of PROCEDURE Read_Font }

BEGIN { Main Program }
  WRITE (CHR (28));
  GOTOXY (15, 0);
  WRITELN ('Character Font Transfer * * * By Timothy S. Smith');
  GOTOXY (0, 2);
  WRITE (CHR (2));
  GOTOXY (27, 23);
  WRITE ('COPYRIGHT 1983 by ON THREE');
  GOTOXY (0, 10);
  WRITE ('[C]onvert Screen Font OR [L]oad Printer Font From Disk ?');
  REPEAT
    READ (KEYBOARD, Response);
  UNTIL (Response = 'c') OR (Response = 'C') OR
        (Response = 'l') OR (Response = 'L') OR
        (Response = CHR (27));
  IF Response = CHR (27) THEN
    EXIT (Program);
  IF (Response = 'l') OR (Response = 'L') THEN
    Load
  ELSE
    BEGIN
      Read_Font;
      WRITE (CHR (28));
      GOTOXY (0, 10);
      WRITELN ('Type [Escape] to quit');
      WRITELN;
      WRITE ('Send Font TO [P]rinter OR [D]isk ?');
      READ (KEYBOARD, Response);
      IF Response = CHR (27) THEN
        EXIT (Program);
      IF (Response = 'd') OR (Response = 'D') THEN
        BEGIN
          Save;
          EXIT (Program)
        END
      ELSE
        Send
    END;
  WRITE (CHR (28));
  GOTOXY (0, 10);
  WRITE ('Test Font (Y/N) ?');
  REPEAT
    READ (KEYBOARD, Response);
  UNTIL (Response = 'y') OR (Response = 'Y') OR
        (Response = 'n') OR (Response = 'N');
  IF (Response='Y') OR (Response='y') THEN
    Test;
  WRITE (CHR (28))
END. { Of PROGRAM Font_Mover }
```

Apple][CharDownLoad

```
10  REM        APPLE DMP CUSTOM
20  REM       CHARACTER FONT LOADER
30  REM       ---------------------
40  REM        BY TIMOTHY S. SMITH
50  REM
60  REM        COPYRIGHT 1983 BY
70  REM         O N    T H R E E
80  REM         APRIL-MAY, 1983
90  REM
100 SL = 1: REM  <<== PRINTER SLOT #
110  REM
120  TEXT:HOME :D$ = CHR$ (4): PRINT D$"NOMON"
130  HTAB 12: PRINT "CUSTOM FONT LOADER"
140  VTAB 22: HTAB 12: PRINT "BY TIMOTHY SMITH"
150  HTAB 7: PRINT "COPYRIGHT 1983 BY ON THREE": VTAB 11: PRINT
160  DIM A(768)
170  INPUT "ENTER FONT FILENAME:";F$
180  IF F$ = "" OR F$ = "?" THEN HOME: PRINT D$"CATALOG": PRINT: GOTO 170
190  HOME
200  REM
210  REM     ** READ FILE OF INTEGER **
220  REM     ** BYTE VALUES FOR FONT **
230  REM
300  PRINT D$"OPEN "F$
310  PRINT D$"READ "F$
320  FOR C = 0 TO 767
330  INPUT A(C)
340  VTAB 12: PRINT "READING =) ";
350 T = INT ((C / 8) + 32): IF T > 96 THEN T = T - 64: INVERSE
360  PRINT  CHR$ (T): NORMAL
370  NEXT C
380  PRINT D$"CLOSE "F$
400  REM
410  REM    ** SEND FONT TO PRINTER **
420  REM
430  HOME: VTAB 12: HTAB 17: FLASH: PRINT "WORKING";: NORMAL
440 P = (49280 + (SL * 16)): REM  <= DATA OUTPUT
450 DR = 49345 + (SL * 256): REM  <= DATA READY STROBE
500  REM
510  REM     PRECEDE FONT WITH
520  REM     PRINTER MODE AND
530  REM     ATTENTION CODES...
540  REM
550  POKE P,24: POKE DR,1
560  POKE P,27: POKE DR,1
570  POKE P,45: POKE DR,1
580  POKE P,27: POKE DR,1
590  POKE P,73: POKE DR,1
600  REM
610  REM      SEND FONT DESCRIPTION
620  REM      BYTES PRECEDED BY
630  REM      1 BYTE = ASCII VALUE
640  REM      1 BYTE = LENGTH OF DESCRIPTION
650  REM      N BYTES = FONT DESCRIPTION
660  REM
670  REM      POKE BYTES TO PRINTER
680  REM      CARD TO AVOID CONFLICT WITH
```

Program listing continued on page 23

Assembling (ON) the ///

by Martin Nichols

Just as promised, this month I will show you how to do some more fascinating things with your ///. Since they are just a random collection of things I have been working on I have called these assembly language routines 'Misc.'.

With this module you will be able to lock and unlock the reset key so that a user program can prevent the computer from being rebooted, just like Visicalc and Apple Writer ///. You will also be able to reboot the system without having to press **CONTROL-RESET.**

The last two procedures in this module allow you to selectively choose the speed of the microprocessor that the Apple /// uses. You can choose either **Slow** or **Fast** and the computer will then operate at either 1 or 2 MhZ. You can use this to 'Slow-Up' those fast listings and cataloges. Like I said last time, this one is for those of you who feel that life is going by too fast.

Many of you can probably see the use of locking and unlocking the RESET key, and possibly even changing the speed to slow things down. But why the heck do we need this reboot thing? Well, I wrote this one because a few buffered printer interface cards loose their buffer when you press CONTROL-RESET.

Say you are printing out a very long listing of a Basic program to your buffered printer. The computer sends the listing to the buffer and the buffer sends it to the printer as fast as the printer can accept it. The listing to the buffer takes only a few seconds and now you want to write a letter so you insert a word processing disk and press CONTROL-RESET. Bam!! The listing that was in the buffer seems to have been erased and you get half a print-out or less.

With this module you can do a reboot without destroying information in your print buffer. Program **listings #1 and #2** show how easy it is to do this from both Basic and Pascal. The assembly language procedures are listed in program **listing #3.** The documentation and test programs combined into one and are listed in program **listing #4.**

To use these routines in your programs, use the Pascal editor to enter the assembly language routines in **listing #3.** Assemble it and name if 'MISC'. The file that the assembler creates will contain the useable assembly language procedures, and its name will be 'MISC.CODE'. To show that this can be used as an invokable module, use the filer to change its name to 'MISC.INV'.

Program **listing #1** is a Pascal Program to show you how easy it is to reboot. **Listing #2** is the Basic implementation. You can use these routines just as they are described in the Basic Documentation and Test program. If you want to use them in Pascal, declare each of the routines external procedures and use the Linker to link them into your Pascal host program.

That's all for now, next time I will answer many readers questions by showing how to format a diskette outside of the System Utilities Disk. Basic and Pascal implementations will be given. Now ANY program can give the user the option of formatting a diskette! ///

Assembling (ON) the ///: Program Listing #1

```
PROGRAM Boot;

{ ************************************************************ }
{ *  Re-Boot (Pascal version)          ----------------------  * }
{ *  ------------------------          |  Copyright 1983 by |  * }
{ *  by Martin Nichols                 |   O N   T H R E E  |  * }
{ *                                    ----------------------  * }
{ *  This program demonstrates how to use the 'MISC' assembly  * }
{ *  language routine ReBoot from within your Pascal programs. * }
{ *                                                            * }
{ *  After compiling this program, use the Linker to link      * }
{ *  the assembly language routine to this Pascal program.     * }
{ *                                                            * }
{ ************************************************************ }

CONST Return = 13;
      Escape = 27;
      Clear_viewport = 28;

VAR Response: CHAR;
    Esc_Ret : SET OF CHAR;

PROCEDURE ReBoot;   { * Here is the routine that allows you to * }
          EXTERNAL; { * reboot without pressing CONTROL-RESET. * }

BEGIN { Main of Boot }
  Response := ' ';
  Esc_Ret := [CHR (Escape), CHR (Return)];
  WRITE (CHR (Clear_viewport));
  GOTOXY (12, 12);
  WRITE ('Press "RETURN" to re-boot the system, ');
  WRITE ('"ESCAPE" to exit');
  REPEAT
    UNITREAD (2,Response,1,,12)          { * Read one character * }
  UNTIL (Response IN Esc_Ret);           { * from the keyboard. * }
  IF (Response = CHR (Return)) THEN
    Reboot
END. { Of PROGRAM Boot }
```

Assembling (ON) the ///: Program Listing #2

```
0   REM ************************************************************
1   REM *  Re-Boot (Basic version)          ----------------------  *
2   REM *  ------------------------         | Copyright 1983 by |  *
3   REM *  by Martin Nichols               |  O N   T H R E E  |  *
4   REM *                                  ----------------------  *
5   REM *  This program demonstrates how to use the 'MISC' invokable *
6   REM *  module from Basic.  Any time you want to re-boot your     *
7   REM *  system without pressing CONTROL-RESET, you can!           *
8   REM *                                                          *
9   REM ************************************************************
10   TEXT:HOME:VPOS=12
20   ON ERR PRINT"Can't find the invokable!!!":END
30   INVOKE"Misc.INV":OFF ERR
40   Esc.Ret$=CHR$(27)+CHR$(13)
50   PRINT USING"$0c";"Press 'RETURN' to re-boot, 'ESCAPE' to exit"
60   GET Response$
70   IF NOT INSTR(Esc.Ret$,Response$) GOTO 60
80   IF Response$=CHR$(13) THEN PERFORM ReBoot
90   HOME
```

Assembling (ON) the ///: Program Listing #3

```
; ************************************************************
; *                                    ----------------------  *
; *  Misc. Utilities:  SpeedBoot(Un)Lock?  | Copyright 1983 by |  *
; *  ---------------------------------     |  O N   T H R E E  |  *
; *  by Martin Nichols                     |  April-May, 1983  |  *
; *                                        ----------------------  *
; *                                                            *
; *  These assembly lanuage routines will enable your Basic or *
; *  Pascal programs to re-boot the system without pressing     *
; *  Control-Reset, Lock and Unlock the Reset key to protect   *
; *  from accidental reboots, and slow down and speed up the   *
; *  speed of the computer.                                    *
; *                                                            *
; *  To use in your Basic programs, assemble these routines using *
; *  the Pascal assembler, and then invoke it as you would any *
; *  other invokable module.                                   *
; *                                                            *
; *  For use in Pascal programs, declare each of these routines *
; *  EXTERNAL PROCEDURES and then use the linker to link them to *
; *  your Pascal host program.                                 *
; *                                                            *
; ************************************************************

        .MACRO  Set             ; This sets specific bits within
        LDA     #%1             ; a byte to 1.
        ORA     %2              ; Use like this -> Set  #80,High
        STA     %2              ; This will set the high bit (#7)
        .ENDM                   ; of the byte High to 1.

        .MACRO  Reset           ; This sets specific bits within
        LDA     #%1^Mask        ; a byte to 0.
        AND     %2              ; Use like this -> Reset  #80,Low
        STA     %2              ; This will set the high bit (#7)
        .ENDM                   ; of the byte Low to 0.

        .MACRO  SOS             ; Macro def for SOS call block
        BRK                     ; Begin SOS call block
        .BYTE   %1              ; call_num
        .WORD   %2              ; parameter_list pointer
        .ENDM                   ; end of macro definition


Close   .EQU    0CC             ; call_num for CLOSE
Mask    .EQU    0FF             ; Used to EOR a byte in Reset above
Envrmt  .EQU    0FFDF           ; Environment register location
Boot    .EQU    0F4EE           ; Monitor entry point for booting


;                       - Procedure ReBoot -
;
; This procedure causes the system to reboot.  Before rebooting,
; it first closes all open files whose file level is greater than or
```

Program listing continued on next page.

```
; equal to the current system level.  Make sure you have finished
; writing to any open file before using this call, as it doesn't
; return.
;
        .PROC   Reboot,0

        JMP     Start

C_List  .BYTE   01              ; one parameter for CLOSE
C_Ref   .BYTE   00              ; close all files that are
                                ; >= the current file level

Start   SOS     Close,C_List    ; Close all open files.
        LDA     #73             ; Do this to make sure we are not
        STA     Envrmt          ; going to jump off into oblivion.
        JMP     Boot            ; No RTS because we won't be
                                ; coming back to the system.

;                   - Procedure ResetLock -
;
; This procedure locks out the Reset key so you can't reboot by
; pressing CONTROL-RESET.  You can use this to idiot-proof your
; programs from accidental rebooting.
;
        .PROC   Reset_Lock,0

        Reset   #10,Envrmt
        RTS

;                   - Procedure ResetUnLock -
;
; This procedure unlocks the Reset key so you can reboot by
; pressing CONTROL-RESET.  Use this just before exiting your
; programs that use the procedure ResetLock above.
;
        .PROC   Reset_Unlock,0

        Set     #10,Envrmt
        RTS

;                     - Procedure Slow -
;
; This procedure cuts the speed of the Apple ///'s 6502 CPU from
; a maximum of 2MhZ to 1MhZ.  This change lasts until the next
; time you reboot, or until you use the Fast procedure below.
; Useful for slowing down things that are going to fast for you
; such as program listings and other quick operations.
;
        .PROC   Slow,0

        Set     #80,Envrmt
        RTS

;                     - Procedure Fast -
;
; This procedure sets the speed of the Apple ///'s 6502 CPU to
; be 2MhZ regardless of what it was.  Use this to speed back up
; the computer after using the Slow procedure above.
;
        .PROC   Fast,0

        Reset   #80,Envrmt
        RTS

        .END                    ;Of assembly
```

Assembling (ON) the ///: Program Listing #4

```
10   REM  ************************************************************
20   REM  *  Misc. Utilities: Documentation and      ------------------  *
30   REM  *                   Test Program          | Copyright 1983 by | *
40   REM  *  by Martin Nichols                      |  O N   T H R E E  | *
50   REM  *  -----------------------------------    |  April-May, 1983  | *
60   REM  *                                         ------------------  *
70   REM  *  This program demonstrates how to use the "MISC" invokable   *
80   REM  *  module from Business Basic.                                 *
90   REM  *                                                              *
100  REM  *  You can now have the computer Re-Boot, Lock and Unlock the  *
110  REM  *  RESET key, and slow down and speed up the microprocessor    *
120  REM  *  from Basic using simple commands.                           *
130  REM  *                                                              *
140  REM  *  To use in your Basic programs, invoke it into memory with   *
150  REM  *  the statement 'INVOKE "MISC.INV"' and then use the routines *
160  REM  *  as they are outlined below.                                 *
170  REM  *                                                              *
180  REM  ************************************************************
185  INVOKE"Misc.INV":REM Load the Invokable Module
190  Screen.Off$=CHR$(14):TEXT
200  Title$=Screen.Off$+"MISC INVOKABLE MODULE Documentation & Test"
210  GOSUB 510:VPOS=4
220  PRINT"Before and Invokable Module can be used, it must be loaded";
230  PRINT" into the system by   the following Command Format:"
240  PRINT:PRINT")INVOKE MISC.INV":PRINT:PRINT"where MISC.INV can be the";
250  PRINT" name of this or any other Invokable Module.":PRINT:GOSUB 500
290  Title$=Screen.Off$+"MISC INVOKABLE MODULE Documentation & Test"
300  GOSUB 510:VPOS=4
310  PRINT:PRINT TAB(8);"Select one of the following options:":PRINT
320  PRINT TAB(10);"1.  Rebooting the system"
330  PRINT TAB(10);"2.  Locking the RESET key"
340  PRINT TAB(10);"3.  Unlocking the RESET key"
350  PRINT TAB(10);"4.  Slowing down the computer"
360  PRINT TAB(10);"5.  Speeding up the computer"
370  PRINT TAB(10);"6.  End":PRINT
380  PRINT TAB(8);"Which option ";:INPUT a$:x=CONV(LEFT$(a$,2))
390  IF x<0 THEN x=0
400  ON x GOTO 1000,2000,3000,4000,5000,420
410  PRINT TAB(8);"Please enter 1, 2, 3, 4, 5 or 6":VPOS= VPOS-2:GOTO 380
420  HOME:PRINT"Bye!":PRINT:END
500  VPOS=24:PRINT USING"76c";"Press any key to Continue";:GET a$:RETURN
510  HOME:PRINT USING"79c";Title$:PRINT:RETURN
1000  REM --- Reboot the system ---
1010  Title$=Screen.Off$+"-- Reboot --":GOSUB 510
1020  VPOS=4:PRINT"This procedure causes the system to reboot.  Before ";
1030  PRINT"rebooting, it first closes  all open files.  To use, insert";
1040  PRINT" a boot disk (Pascal, Apple Writer ///, etc.)    into the ";
1050  PRINT"internal disk drive and issue the following statement:":PRINT
1060  PRINT")PERFORM Reboot":PRINT
1070  PRINT"and the system will reboot.":PRINT
1080  PRINT"To actually use, exit this program and enter the above ";
1090  PRINT"statement to reboot."
1099  GOSUB 500:GOTO 290
2000  REM --- Lock the RESET key ---
2010  Title$=Screen.Off$+"-- Locking the RESET key --":GOSUB 510
2020  VPOS=4:PRINT"This procedure locks out the RESET key so you can't ";
2030  PRINT"reboot by pressing          CONTROL-RESET.  You can add this";
2040  PRINT" to your programs as a method of protection     against";
2050  PRINT" accidental rebooting.":PRINT
2060  PRINT"To use, enter the following statement:":PRINT
2070  PRINT")PERFORM ResetLock":PRINT
2080  PERFORM ResetLock
2090  PRINT"We have just locked out the RESET key.  To test it out, ";
2100  PRINT"try to reboot the       system by pressing CONTROL-RESET."
2110  PRINT:PRINT"When you are convinced that you can't reboot..."
2120  GOSUB 500:VPOS=16:PRINT
2130  PRINT"See, I told you so..."
2140  GOSUB 500:GOTO 290
3000  REM --- UnLock the RESET key ---
3010  Title$=Screen.Off$+"-- UnLocking the RESET key --":GOSUB 510
3020  VPOS=4:PRINT"This procedure unlocks the RESET key so you can ";
3030  PRINT"reboot by pressing CONTROL-RESETYou can add this to";
3040  PRINT" your programs as a means of reversing the protection";
3050  PRINT" given  by the ResetLock statement.":PRINT
3060  PRINT"To use, enter the following statement:":PRINT
3070  PRINT")PERFORM ResetUnLock":PRINT
3080  PERFORM ResetUnLock
3090  PRINT"We have just unlocked the RESET key.  Take my word for it, ";
3100  PRINT"if you try to test itby pressing CONTROL-RESET you will ";
3110  PRINT"reboot the system."
3120  GOSUB 500:GOTO 290
4000  REM --- Slow up the /// ---
4010  Title$=Screen.Off$+"-- Slowing down the Apple /// --":GOSUB 510
4020  VPOS=4:PRINT"This procedure cuts the speed of the Apple ///'s ";
4030  PRINT"6502 CPU from a maximum of 2MhZto 1MhZ.  This change lasts";
4040  PRINT" until the next time you reboot, or until you use theFast ";
4050  PRINT"procedure included in this invokable module.":PRINT
4060  PRINT"To use, enter the following statement:":PRINT
4070  PRINT")PERFORM Slow":PRINT
4080  PERFORM Slow
4090  PRINT"We have just slowed down the microprocessor.  Everything ";
4100  PRINT"should appear slower   now.  All operations will take about";
4110  PRINT" 50% more time to complete."
4120  GOSUB 500:GOTO 290
5000  REM --- Speed up the /// ---
5010  Title$=Screen.Off$+"-- Speeding up the Apple /// --":GOSUB 510
5020  VPOS=4:PRINT"This procedure sets the speed of the Apple ///'s ";
5030  PRINT"6502 CPU to a maximum of 2MhZ  from 1MhZ.  This change lasts";
5040  PRINT" until the next time you reboot, or until you use  the ";
5050  PRINT"Slow procedure included in this invokable module.":PRINT
5060  PRINT"To use, enter the following statement:":PRINT
5070  PRINT")PERFORM Fast":PRINT
5080  PERFORM Fast
5090  PRINT"We have just speeded up the microprocessor.  Everything ";
5100  PRINT"should appear faster    now.  All operation should be at";
5110  PRINT" normal speed."
5120  GOSUB 500:GOTO 290
```

DOS File List: Continued . . .

```
BEGIN { MAIN of List_DOS_Directory }
  Set_Out_device;
  Line_count := 4;
  With Dosdir[0] DO
    BEGIN
      Dfkind := Volinfo;
      Dnumentries := 0;
      Dunitnum :=0
    END;
  Catalog;
($IOCHECK- )
  CLOSE (Device, LOCK);
($IOCHECK+ )
  Trap_IO_error
END; { Of PROCEDURE List_DOS_Directory }

BEGIN
  { This is the initialization, which occurs }
  { before the host program is executed. }
END. { Of UNIT DOS_List_Stuff }
```

Products Received

The products outlined below have been received by ON THREE for the purpose of review. Some have been reviewed in the past and many will be reviewed in the future. The products have all been given the ON THREE 'stamp of approval'. This is only an indication that a product works as advertised and is not an endorsement of the product by ON THREE.

PFS: FILE & REPORT

With PFS: File you can create a file, search and update any item or group of items in the file, and print sorted information. Information management at its best, these programs are extremely easy to use.

All PFS products are designed so that a novice can master them in less than an hour. Reviewed in the January issue of ON THREE, these programs received an A- and a B- respectively.

Available from most authorized Apple dealers, these programs are made by Software Publishing Corporation and are priced at $175 and $125 respectively, for the Apple ///.

Software Publishing Corporation, 1901 Landings Drive, Mountain View, California 94043. (415) 962-8910.

QUICK & EASY DATA MASTER

Quick & Easy Data Master is a program that creates custom applications software and report forms designed to your specifications. This package creates an unprotected Business Basic data base program as per your specifications.

The ideal data base program is one that you can design exactly the way you want it: prompts, edits, error messages, headers, titles, computed data, interactive files, report forms, etc. to your specifications. You design it and Quick & Easy will create it for you.

Intended for the more serious computer user who knows how to program in Basic, this package is not very hard to use, but it does require some thought. Sold by Advanced Software Technology, Inc., it is priced at $69.95. Reviewed in this issue, it gets a C.

Advanced Software Technology, Inc., 7899 Mastin Drive, Overland Park, Kansas 66204. (913) 648-4442.

CRITICAL PATH SCHEDULING

If you are involved in project management and tired of the hassels of project scheduling, the Critical Path Scheduling System is for you! It is a management tool for defining and analyzing the overall concepts of a project and provides a powerful method for scheduling the many tasks necessary to complete the project ON TIME AT THE LOWEST POSSIBLE COST.

Armed with the information that this system provides, the manager is better prepared to make decisions regarding the impact any task will have on the project and permits him to be instrumental in guiding the project rather than just monitoring its progress.

This is a very 'User Friendly' system, and it has a good tutorial/user manual. Comprehensive reports make a manager's life a lot easier. Developed by Great Divide Software, it has a suggested retail price of $495. Reviewed in this issue, CRITICAL PATH SCHEDULING gets a B.

GL-PLUS

To many managers, accounting and the preparation of financial reports are time consuming chores that have to be struggled through. But now, at last, accounting can be simplified.

GL-PLUS is an accounting system designed for the Apple /// computer. It is a flexible, easy to use, journal-based General Ledger system. The computer and GL-PLUS combine to provide you with a tool. A tool to make your accounting chores easier. GL-PLUS automatically guides you through entries and then automatically sorts and posts them.

Report preparation is a "snap"with GL-PLUS. You select the report you wish and the rest is done automatically. GL-PLUS includes a PLUS. The PLUS is a built-in accounts receivable and accounts payable capability that can be implemented anytime you desire.

Another 'User Friendly' system, flexible reporting and ease of use make an excellent accounting package. Developed by Great Divide Software, it has a suggested retail price of $495. To be reviewed in the June-July issue.

Great Divide Software, Inc., 8060 West Woodard Drive, Lakewood, Colorado 80227. (303) 337-0383.

PKASO ///

The PKASO /// printer interface system is a hardware and software device that allows the Apple /// to operate with just about any dot matrix printer in both native and emulation mode. It gives the user the option of printing text, graphics or even screen dumps on your parallel printer.

A very nice printer interface, the system has a suggested retail price of $205. Reviewed in this issue, the PKASO /// printer interface gets an A.

Interactive Structures, Inc., P.O. Box 404, Bala Cynwyd, Pennsylvania 92123. (215) 667-1713.

MICROTEK Dumpling-GX & Dumpling Spooler & Alpine Printer Driver

The MICROTEK printer interface cards used with the Alpine printer driver allow the Apple /// to connect with a variety of parallel printers. In particular, the MICROTEK Dumpling Spooler interface allows you to dump vast quantities of data into the interface buffer for later printing.

Combined with the Alpine Apple /// interface software, the MICROTEK interface cards allow the printing of text, graphics and screen dumps on your parallel printer. It even works in emulation mode. To be reviewed in the June-July issue.

MICROTEK, Inc., 9514 Chesapeake Drive, San Diego, California 92123. (619) 569-0900. Alpine Computing, Inc., 851 North Main, Logan, Utah 84321. (801) 752-6432.

Continued on page 9

Word Processing Language

by Bob Consorti

WPL is a fantastic tool for Apple Writer /// users. Unfortunately, very little is known about WPL and what it can do. ON THREE will periodically publish WPL programs and more importantly, the instructions on how to use them. The information in this column will serve as an introduction and tutorial to the language, so if you have written WPL programs we want to hear from you!

What is WPL and what can it do? Well, as stated in the manual, WPL's purpose is to make **automated word processing** possible. Many of the things that you do with word processing such as typing in form letters, reports and other repetitive tasks can be done easier with WPL.

Since one of the greatest purposes of computers is to save us time, it was inevitable that WPL came to be. Since it can and will **save you time,** it's unfortunate that so little information is available for it.

It all sounds good, right? Well, now we will discuss just how WPL can make your work a little easier. To follow along, you must have an Apple Writer /// Master disk in the internal disk drive and the instruction manual (not the Product Training Pak!).

Type [P] (Control-P), and then 'PD.CONSOLE'. Next type [P] 'DO .D1/AUTOLETTER'. We will be printing letters so to not waste paper, we will print it on the screen. That's the reason for the first line. The next line tells the computer to execute the WPL program named 'AUTOLETTER' on the disk in the internal disk drive.

Your disk drive should come on a number of times and five letters will be printed. What is going on is this: The WPL program is taking a form letter (from the file '.D1/FORMLETTER') and the names and addresses from the file '.D1/ADDRS' and combining the information to create form letters. Thus if you have a large number of letters with the same general information that needs to be sent out, WPL can extract the names, addresses and other pertinent information from the address file and automatically print out a number of personalized letters.

How does it do it? That's a bit complicated but I think I can explain it. A WPL program is a standard Apple Writer /// text file that consists mainly of normal Apple Writer /// commands. A WPL program thus has all of Apple Writer ///'s search and replace, printing and formatting, loading and saving, insertion and deletion, etc. commands. For those of you with a little programming expertise, WPL also offers integer and string variables. Even some simple comparison statements and subroutines are available.

As its name implies, WPL is a true programming language. With it you can really automate many of those complicated and tiresome chores. The form letter example included on the Apple Writer /// program disk is but one of many things possible with WPL. For the next few paragraphs, I will discuss the basics of WPL and give you examples of simple WPL programs.

This is going to be an interactive tutorial, so you will have to do a **little work.** Start by opening your instruction manual to page 71 and start reading. On page 72 you will find something that may clarify things a little. If you have ever used embedded text formatting commands when printing out a text file, you have used WPL!

In other words, if you used the '.CJ' command to center a line of text while printing (or any other formatting command) you were using WPL. Thus all of the '[P] Print/Program Commands:' options are caried out by WPL. On page 72 you will also see how to use WPL commands from within a WPL program.

To automate a process using WPL, all you have to do is write down all of the keystrokes you use in that process and convert those keystrokes into a WPL program. Commands begin with some control character (CONTROL P to print, CONTROL F to find/replace . . .), thus to initiate a command from within a WPL program we must somehow tell the computer that we want a command processed.

Using WPL we just enter the CONTROL characters as normal letters and follow them with the other necessary keystrokes. So to set the left margin using WPL we enter the following statement: 'PLM 10'. One of the quirks of WPL is that commands have to be preceded by one or more spaces. Thus press the TAB key once before entering the command 'PLM 10'.

You should now have 14 characters in your file. Save it to a blank disk with the command '[S].D2/TEST.1'. Now that it is saved on a diskette you can execute it by typing in '[P]DO .D2/TEST.1'. This is the "DO" command and it is the way to execute a WPL program.

After pressing RETURN, the "DO" command will execute the specified WPL program from disk. In our example, the program changed the left margin of the print format to 10. To check this out, type '[P]?' to get into the Print/Program Commands: options. The left margin option should now be 10. To make sure that it wasn't 10 to start with, change it to say, 99 with the command 'LM 99'. Now press RETURN once to get out of this menu and execute the WPL program again with the line '[P]DO .D2/TEST.1'. Finally, check the left margin value and it will be 10. Woila! The WPL program did it.

Mindless, huh? Well, say you had one hundred letters to be printed, each with a different left margin. It would be a heck of a job to manually change the margin and print out each file. WPL offers a much easier way.

Type in the following paragraph and save it under the pathname '.D2/LETTER'.

This is a short line of text to illustrate one of the many uses of WPL in letter and general document preparation. When the WPL program is executed, each time you press RETURN this document will be printed with the left margin 5 spaces greater that the last time.

Say that we want the above paragraph printed 7 times. The first time we want the left margin set to 0, the next time set it to 10, and the next 20. . . . It would take some time if we had to do it manually. The WPL program that does all this is below. Clear memory and type in the below WPL program. Remember to press the TAB key before each line.

```
 NY
 PND
 PPD.CONSOLE
 L.D2/LETTER
 PPR
 PLM0
 PNP
 PIN  Press RETURN To Continue
 PPR
 PLM10
 PNP
 PIN  Press RETURN To Continue
 PPR
 PLM20
 PNP
 PIN  Press RETURN To Continue
```

```
PPR
PLM30
PNP
PIN  Press RETURN To Continue
PPR
PLM40
PNP
PIN  Press RETURN To Continue
PPR
PLM50
PNP
PIN  Press RETURN To Continue
PPR
PLM60
PNP
PIN  Done!! Press RETURN To Go Back To The Editor
NY
```

Before saving it, look at the lines 'PPR'. There should be 7 of them. After the 'R' in each of them you must insert a control character. This is the code that causes Apple Writer /// to clear the screen. If not done, the text display will get cluttered up and you won't know what is going on. After each 'R' type 'CONTROL OPEN APPLE \'. Just hold the CONTROL and OPEN APPLE keys down and type the backslash ("\") key. An inverse (black on white) backslash should appear. When you are done save it with the pathname '.D2/TEST.2'.

If you are all done, execute the program with the command '[P]DO .D2/TEST.2'. The paragraph will be printed 7 times, each one with a left margin 10 spaces greater that the last. You did it! Now the question is how?

Going through the WPL program line by line we will first see what the individual lines do and then will be able to look at the program as a whole. The first line is 'NY'. You should be able to see what this line does now. Remember, a WPL command is just the same as a normal Apple Writer /// control command. So the WPL line 'NY' becomes [N]Y. What happens when you press control N and then Y? That's right! Memory is erased. We should all see now that this line clears out memory. You should always use it because you never know what will be in memory when you start a WPL program.

All right, that wasn't too hard. How about the next line, 'PND'? This should correspond to [P]ND. Try it!, type CONTROL P and then ND. Don't get scared, your computer didn't die - it just went into the twilight zone. The command 'PND' shuts off the screen. It means No Display. To turn it back on, type in [P]YD for Yes Display.

The line 'L.D2/LETTER' in WPL translates to [L].D2/LETTER in normal Apple Writer /// notation. This loads the file LETTER from the disk in the second disk drive into memory. Once loaded you can do anything you want with it.

The next line is PPR (Control-Backslash). This is the clear screen command. Unless you want a mess on your screen use this command before you print anything. Since we want the paragraphs printed on the screen and not on your printer we use the line 'PPD.CONSOLE'. This translates to [P]PD.CONSOLE which sends the printed output to the screen.

The line 'PLM0' sets the left margin to 0. Thus all printed lines will start at the leftmost column. The Apple Writer /// translation is [P]LM0. Once we have set the margin we can print the paragraph with the line 'PNP'. This translates into [P]NP which simply instructs Apple Writer /// to begin printing the text in memory.

These lines printed the first version of our paragraph. Since we may want to wait a second or so to admire our work we include the line 'PIN Press RETURN To Continue'. This line simply waits for us to press RETURN.

The WPL program has now printed our first paragraph and is waiting for the user to press RETURN so it can go on and print the other paragraphs. Once the user presses RETURN, the program goes on and prints the other 6 paragraphs.

Once it is done printing the paragraphs, the line 'PIN Done!! Press RETURN To Go Back To The Editor' waits for the user to press RETURN and the next line 'NY' clears out memory. Congratulations, you just wrote (and hopefully **understood**) your first true WPL program.

The PIN command can be used outside of a WPL program. Just type [P] and then enter 'IN Press RETURN To Continue'. The screen display will show 'Press RETURN To Continue'. Do that and you will be back in the editor.

Some of you may be wondering if there is an even easier way of doing this. Looking at the WPL program you will see that there are seven sets of very similar statements. The basic structure of these lines are below:

```
PPR
PLM*
PNP
PIN  Press RETURN To Continue
```

If you replace the asterisk with the numbers 0 through 6, you will get the same WPL program lines as above. The question is, is there a way to do that from inside a WPL program? The answer is an emphatic **YES!**

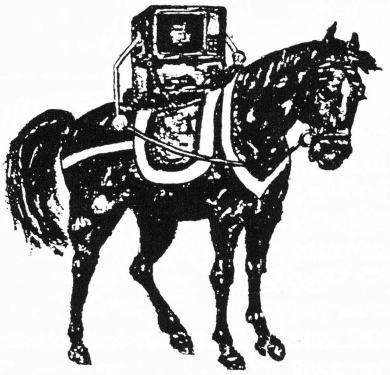
To actually do it we must learn a little more about WPL. Well, let me show you the updated program and then describe it. Our 'Better, Faster, Smarter' WPL program is below.

```
Start   PGO Begin

PrintIt PPR
        PLM   +10
        PNP
        PIN   $A
        PRT

Begin   NY
        PND
        PPD.CONSOLE
        L.D2/LETTER
        PLM0
        PLM-10
        PAS   Press RETURN To Continue =$A
        PSX   6
Loop    PSR   PrintIt
        PSX   -1
        PGO   Loop
        PAS   Done!! Press RETURN To Go Back To The Editor =$A
        PSR   PrintIt
        NY
        PQT
```

In the above WPL program we use subroutines, error flagging and detection, integer and string variables, and even a simple program loop. Before you type it in, remember about inverse backslash. Now how does it work? Well, all the commands are described on pages 78-79, 83 and 88-90. If you read over those sections of the instruction manual you should understand what's going on. Next time we will discuss it further. We will also enhance the HELP instruction with a couple of new options! Until then, hit the books and start learning (and using) WPL! ///

/// to the Max

by Al Evans



Tame the Running Horses

Here it is, Apple /// users, the column for crazies, the forum for people who want their computer to do MAGIC and don't care about the obstacles. We're not talking about stuff that can easily be modified to work on a RadShack or an IBM or even an Apple][. We're not even necessarily talking about code that's legal according to SOS. We're talking about /// **to the Max** — tricks an Apple /// can perform which will astound the lesser computer.

There are those who preach device-independent code — programs which will work on any computer which runs, for example, UCSD Pascal or FORTRAN 77. That's great for the majority of applications. But this column is about **maximum performance.** To get the best performance from **any** computer, you must use device-dependent techniques which take maximum advantage of its special features and abilities. And the Apple /// has special features and abilities which make it capable of running rings around any computer in its class. So this column will be about the hardware and operating system of the /// and software which makes special use of them, whether in BASIC, Pascal, or assembly language.

For an easy introduction to our general orientation, we'll consider the first question asked in our initial column: How did they make those horses run? This will be an interactive investigation. You'll need an Apple /// of any configuration, an Apple Business Basic Disk, and a System Demonstration Disk. The following instructions assume that you're using a system with two floppy-disk drives. If not, just use the built-in drive and substitute ".D1" wherever ".D2" appears.

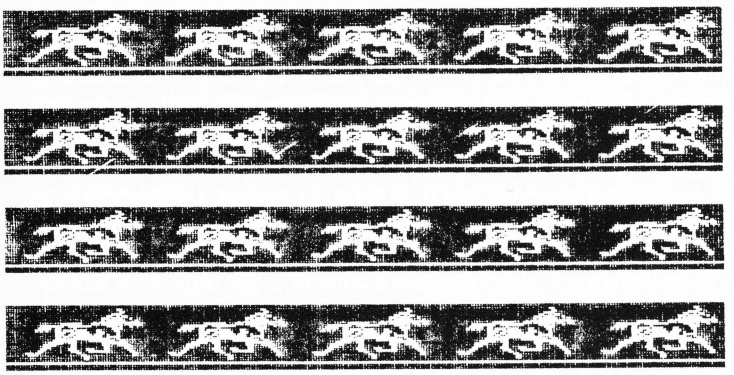
First boot up your /// in Business Basic, and put the "SYSTEM DEMONSTRATION" disk in drive 2. Don't boot up on the demo disk; there's no way to get out of the demonstration, which I'm sure you've already seen.

Now type "LOAD .D2/SHOW" (and press <RETURN>, of course). Delete lines 0-8999 and lines 10000-24030. Also delete line 9125 and line 9190. The part of the program which is left is the part that makes the horses run. Only fifteen lines.

Type "INVOKE .D2/HORSES.INV" and "RUN." You should see exactly the same horserace shown in the demo program. That fifteen lines of BASIC, with the three external procedures HINIT, HFRAME(%I), and HSCROLL, is doing the whole thing. Unfortunately, these three procedures are written in assembly language, and there's no simple way to find out how they work. Fortunately, however, we can find out exactly **what they** do from BASIC, and that's nearly as good. After all, we can always write our own programs to **do** the same thing.

Anyway, your screen should now look about like Figure 1, all covered with horses. We need to get rid of all of them but the top line in order to continue the exploration. Press <ESCAPE> to put your /// in the escape mode, and use the "up-arrow" and "left-arrow" keys to move the cursor just under the lower left-hand corner of top row of horses. You won't be able to see the cursor when it's there — this is interesting and important, and we'll come back to it later. For now, just press "T" to set the top of the viewport below the horses we're keeping, then move the cursor to the bottom of the screen and scroll the other three rows of horses offscreen.

Figure #1



Now move the cursor back directly beneath the row of horses, press the space bar to leave escape mode, and type:

```
FOR I=0 TO 3:FOR J=0 TO 7:PRINT CHR$(128+8*I+J);:NEXT:
PRINT:NEXT
```

Congratulations, you just drew your own horse. Your screen should now look like Figure 2. Those of you who already know why that happened can take a break; go on and skip the next paragraph.

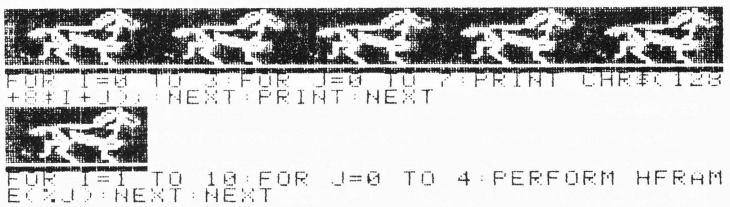
Figure #2

Most of the rest of you do already know that the Apple /// system character set is software-defined (like almost everything else in the system) and can be changed at any time. In particular, characters 0 to 31 [CHR$(0) to CHR$(31)] are the control codes, and are not normally printed on the screen. However, any of them can be turned into a printable character by adding 128 to its ASCII code [CHR$(0+128) to CHR$(31+128)]. This is just what we've done in the line above — printed out the current representations of the control codes in four lines of 8 characters each. **The control characters have been turned into horses!!** This is one of the major keys to fast animation on the Apple ///.

Press <ESCAPE> again, move the cursor to the line immediately below the horse you just drew, and reset the top of the viewport (press "T") to keep it on the screen. Scientifically speaking, we **know** where that horse came from — we can't be completely sure the others didn't appear by magic. Now type the following line (see Figure 3):

FOR I=1 TO 10:FOR J= 0 TO 4:PERFORM HFRAME(%J):NEXT:NEXT

Figure #3



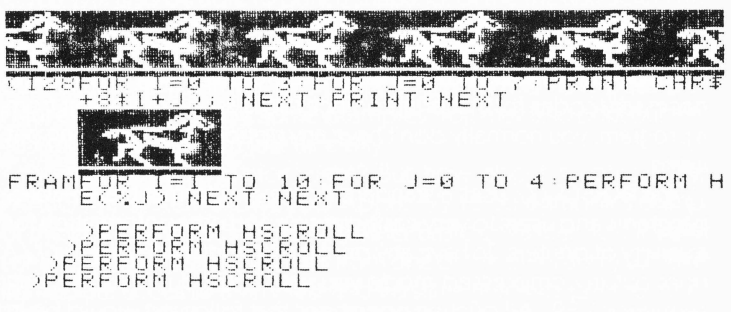
See the horses run! But what's happening now? We're not even writing anything to the screen. Not only that, but the horse we drew ran right along with the others — HFRAME(%J), whatever it is, couldn't possibly know about our own personal horse! Once again, this is a major key to Apple /// animation.

What happened is that the definitions of those control characters were changed "on the fly" under software control. The animation is almost free — not one character had to be re-drawn. Moving 6 horses was as fast as moving 1. If there could have been a hundred little horses on the screen, they all would have moved just like those 6. To be specific, the programming technique used is a "partial character set upload" — see the **Apple /// Standard Device Drivers Manual, p. 71.**

So that's how the horses run. But wait a minute — they also ran **across** the screen. And remember when the cursor disappeared, when we first set the top of the viewport?

If there's any one of you who hasn't figured out by now that "PERFORM HSCROLL" must be the proper command, you really should pay closer attention. Go ahead and type it in a few times, then we'll analyze what is happening. Figure 4 shows the screen after 4 PERFORM HSCROLL's

Figure #4



Hmmm . . . The horses just slid across the screen. So did all the text we put on it! Type "GOTO 9030" and you can see the whole horse-and-text race. The horses and letters that disappeared on the right re-appeared on the left — looks like this is a circular device of some kind. But the color blocks just stood still. Looks as though the horses and text are in one part of memory, and the background and foreground colors are in another.

And that is precisely the case. The screen memory of the Apple /// is in two places, both in the system bank ("S-bank"). The characters on the 40-column screen (or the odd-numbered characters on the 80 column screen) are in memory locations $400-$7FF (1024 to 2047 for those who speak decimal). The foreground and background colors in the 40-column mode (or the even-numbered characters on the 80-column screen) are in memory locations $800-$BFF (2048-3071). HSCROLL is moving the former, shifting each row one column to the right with the character shifted off the screen going into the vacated column on the left, without changing the latter.

In the 40-column color text mode, each byte in the screen at $800-$BFF is interpreted as a background color (the low nybble) and a foreground color (the high nybble). If both of these nybbles are set to the same value, anything on the foreground screen will "hide" behind the background. This explains the disappearing cursor; both nybbles were set to 0 (black).

This is a good example of a device-dependent technique which is not even "within the system". As far as I know, there is no way these screens can be manipulated separately from a high-level language. For those of you who are interested, the memory maps are the same as for Apple][text screens 1 and 2. Apple][assembly-language routines should be fairly easy to adapt for your own nefarious purposes.

What can be done with all this? I don't know about you, but it's got my wife and me working on a game, **Cap'n Magneto.** So far, we've got a background of 256 x 128 squares that scrolls in all 4 directions with wraparound and an animated superhero who is normally a spaceman with blasting jetpaks but turns into a fluttering butterfly on occasion. The sky's the limit!

And so, off into the sunset, jetpaks blasting. Next month we'll talk about the keyboard map and how to change it. After that — well, let me know what you want to do with your Apple ///. In particular, send me your own findings and let me spread them around — I'll give you full credit, of course. There are lots of completely undocumented bits and pieces in the ///, things Apple doesn't think we're ready for. So it was with the][, back in the dark ages. But users, working together and trading ideas, had figured the][out long before the "real" technical manual was ever published. We can do the same. ///

Apple][CharDownLoad: Continued . . .

```
690  REM      <CTRL> CHARACTERS ACTION.
700  REM
710  REM      SO <CTRL> [ I ] IS NOT
720  REM      INTERCEPTED.
730  REM
740  FOR C = 32 TO 126
750  POKE P,C: POKE DR,1
760  POKE P,8: POKE DR,1
770  FOR B = 0 TO 7
780  POKE P,A((C - 32) * 8 + B): POKE DR,1
790  NEXT B
800  NEXT C
810  POKE P,4: POKE DR,1
820  HOME: VTAB 12: PRINT "TEST (Y/N) ?";: GET A$
830  IF A$ < > "Y" THEN END
900  REM
910  REM      ** TEST FONT **
920  REM      SEND CUSTOM FONT
930  REM        SELECT CODE
940   REM
950  REM       PRINT FONT
960  REM
970  REM       SEND STANDARD FONT
980  REM       SELECT CODE
990  REM
1000 HOME
1010 PRINT: PRINT D$"PR#1"
1020 PRINT CHR$ (27)"'";
1030 FOR X = 32 TO 126
1040 IF X / 60 = INT (X / 60) THEN PRINT
1050 PRINT CHR$ (X);
1060 NEXT X
1070 PRINT CHR$ (27)"$"
1080 PRINT
1090 PRINT D$"PR#0"
1100 HOME
```

REVIEW ON: Everything

by Bob Consorti

PKASO /// Printer Interface

The PKASO /// printer interface system is a hardware and software device that allows the Apple /// to operate with just about **any dot matrix printer** in both native and emulation mode. It gives the user the option of printing text, graphics or even screen dumps on your parallel printer.

It will connect your Apple /// to the following printers: Apple DMP; Epson MX-70, MX-80, MX-100; NEX PC8023A/C; C.Itoh 8510 Pro/Writer; Integral Data Systems 460, 560, Prisms, Color Prisms; Okidata 82A, 83A (plus the new 9x series); or Centronics 739, 122 printers.

The PKASO system consists of the hardware interface that allows you to connect your /// to most any parallel dot matrix printer, the necessary cabling, the software device drivers that let the interface card work with the ///, invokable modules to print out the graphics screen and a set of instruction manuals to tell you how to operate the system.

First we are going to take a look at the interface card. Very simply, it is the same interface card the the Apple][uses. What makes it special is the device driver that connects it to the ///'s world and the appropriate invokable modules that allow you to print out the graphics screen. After installing the device driver on your system diskettes you can use a variety of new commands to easily control your printer features from within any program or language system.

The invokable module (or linkable, if you use Pascal) that prints out the graphic screens is very flexible and can print out any of the Apple /// graphics modes in three different sizes. You can even rotate the picture to print out sideways if your printer doesn't have a wide enough carriage to print out one of the larger sizes.

One of the nicest features of this system is the ability to print out in **color** if you have a color printer! Your IDS Prism can now print out your color graphics screens with this package. Many of you know that the /// has a changeable character font and that you can re-define the format of the characters that you see on the text screen. You have probally seen some of the different fonts, Roman, Byte, Apple etc. With the PKASO /// system you can have your printer print out text in the same font style that your /// uses!

You can even dump a copy of the text screen to the printer. Anything that is on the current text screen will be exactly duplicated on the printer. You can dump a 40 or 80 column screen with amazing simplicity.

If you are printing text or graphics and your printer runs out of paper, the PKASO card will put a flashing message on the bottom right hand corner of the text screen. This is non-destructive, that is to say, anything that was there before will be returned whenever you put more paper in the printer. Also, if your printer is off and you try to use it you will get a printer off line message in the same area.

One note of concern: Short of turning on the printer or putting more paper in, there is no way to get out of the print operation if the printer is off-line or out of paper. While you can get out of a text or graphics dump by pressing the ALPHA-LOCK key, there is no similar exit from either of these conditions. A call to Interactive Structures reveals that they are working on a fix to this slight problem.

This brings me to my next point. Interactive Structures, the maker of the PKASO /// card has one of the **best customer support** policies I have seen. Everyone there seems to know what is going on and are more than helpful with any problems you may be having. I sometimes wonder if their support has tied up too many people, but it does give the company a well deserved good reputation.

One of the novelties of the PKASO system is that the instruction manuals were printed with an MX-80 printer controlled by a PKASO board. The neat part is that it looks quite good! The manual that we will discuss is the PKASO /// Printer Driver for Apple /// and Matrix Printers Users Manual.

This booklet tells you everything you need to know about the operation of the PKASO driver and associated software that are needed in order for the PKASO board to work with the Apple ///. It doesn't assume anything about the intelligence of the user so it gives complete information on how to first install the driver using the System Configuration Program and then on how to use it.

Since the SCP isn't easy for first time users, this is a very nice touch that other packages should have. After installing the driver, the manual shows you how to use this new driver from both Basic and Pascal. You can change the size of the characters you print by simply entering a new size with the size command.

The PKASO /// interface has circumvented many of the problems of using the features (changing text sizes, column widths, setting tab stops, etc.) of a particular printer by creating a system that more or less standardizes these procedures.

Instead of remembering strange character sequences to send to each different printer to have it print in say, compressed characters, with the PKASO /// installed you will be able to just type one command sequence for all. Since this may not be easy to see, the following example may help.

Normally to have an Epson print in compressed mode (17.16 characters per inch, others are different) you must send it a CONTROL-O. Other sizes can be achieved by sending different control codes. These codes are not standard among printers, thus what works on an Epson will not usually work on a NEC or others.

This means that if a program wants to print out data in a different size format, it must know what printer it is working on and the necessary codes for that printer. Since this is asking quite a bit from a program, you normally don't have any options when printing out items.

The PKASO /// uses a standard command format that enables programs and users to very easily set size and other information for a variety of printers. To have any printer that is installed on your /// print out in compressed mode you would send it the following sequence '~E2'. All printing operations that followed would be in compressed mode.

Similarly you could have the information printed using the following commands '~E0', '~E1', '~E2', '~E8', '~E9', '~E10'. Respectively, this would cause characters to be printed with 10, 12, 16.5, 5, 6 and 8.25 characters per inch.

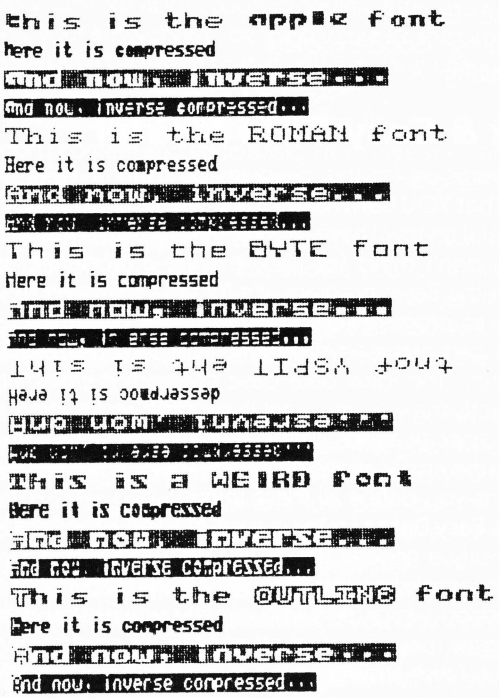
As you can see, remembering '~Ex' is a lot easier than 6 or more control combinations for printing in different sizes. There is one thing that you should be aware of, if your printer doesn't normally print with say, 6 characters per inch, the PKASO /// does not let your printer do that. If you specify a size that does not exist on your printer, the size actually printed will either not change or will be the closest size that your printer does support to the one you asked for.

Similarly, you can easily set up to 16 tab stops and clear them.

You can also adjust the vertical spacing to be any increment and even issue line and form feeds with ease.

One of the nicest features of the PKASO /// is the ability to **change the font** of the printer from software. You can now print out text with the font style you are using on the screen! Printing out text in fancy gothic script is now very easy. You can also choose the inverse screen font and get characters printed in reverse, white on black. **Figure #1** shows some of the different combinations possible.

Figure #1



You will also be able to print a snapshot of the the text screen in either normal or inverse (white on black). Just send ' P2' and the 80 column text screen will be printed out on your printer. An exact duplication of whatever is on your text screen is now possible! **Figures #2 and #3** give some examples. You may also print out the 40 column text screen just as easily.

Printing out text is nice, right? But how about graphics! The PKASO /// system allows your Apple /// to do a graphics dump of the Hi-Res graphics screen in either Basic or Pascal. Just invoke (or link) the necessary routines from one of the disks and you have immediate acces to a very fast and versatile graphics print routine.

You can have a print out of any of the graphic modes that the Apple /// supports in a variety of sizes and shades. Another very nice feature is the ability to center the picture in the middle of the page by setting the margin. You can also stop a print-out by pressing the ALPHA LOCK key twice. This is in case you discover you're printing in the wrong mode and don't want to wait for the whole graphics screen to be printed.

If you are fortunate enough to have access to a IDS color Prism the PKASO /// will do a Hi-Res dump in **full color.** This is something to see! All those pie charts and other graphics screens that look great on a color monitor look even better on paper. Since I can't give you a full color picture in the magazine (yet) you will have to be satisfied with the black and white ones that follow. **Figures #4 and #5** show these graphics dumps.

The instruction manuals show you how to use the PKASO /// system with different software packages. Apple][emulation mode, Apple Writer ///, Apple Business Graphics, Basic, Pascal and Visicalc are all supported. Complete documentation is provided on exactly how to use the PKASO /// system from all these software products.

Figure #2

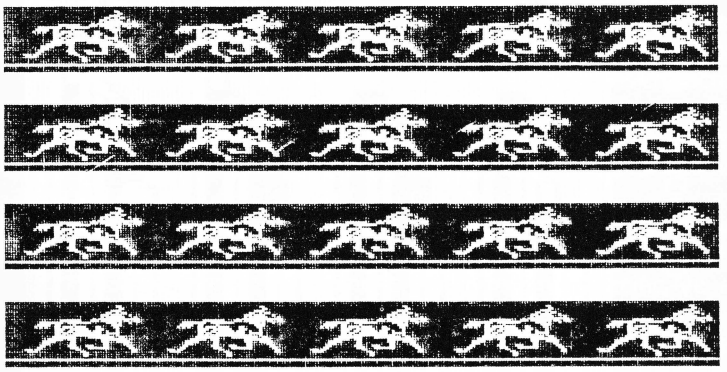


Figure #3

```
PROFILE2        (05/04/83) V2

 TYPE   BLKS  NAME           MODIFIED TIME  CREATED  TIME   EOF
*CAT    00002 SYSTEM         03/12/83 00:57 05/04/83 21:51  1024
*PASDTA 00003 OPCODES.6502   03/07/83 21:38 05/04/83 21:53  720
*PASDTA 00008 ERRORS.6502    03/07/83 21:39 05/04/83 21:53  3570
*PASCOD 00124 UTIL.CODE      03/07/83 21:39 05/04/83 21:53  62976
*CAT    00002 SOS            03/12/83 00:57 05/04/83 21:53  1024
*CAT    00002 VISICALC       03/12/83 00:57 05/04/83 21:57  1024
*CAT    00002 BASIC          03/12/83 00:57 05/04/83 21:59  1024
*CAT    00001 PASCAL         03/12/83 00:57 05/04/83 22:05  512
*CAT    00002 ARTICLES       03/12/83 00:57 05/04/83 22:12  1024
*CAT    00001 D.O.M.         03/12/83 00:57 05/04/83 22:12  512
*CAT    00009 LETTERS        03/12/83 00:57 05/04/83 22:16  4608
*CAT    00001 PFS            05/04/83 22:34 05/04/83 22:34  512
*CAT    00002 SUBS           05/04/83 22:37 05/04/83 22:37  1024

BLOCKS FREE: 1292  BLOCKS USED: 8436  TOTAL BLOCKS: 9728
```

Included on the disks that come with the package are excellent demonstration programs in both Basic and Pascal. They guide the user into a good understanding of just what the PKASO /// system can do. The manuals that come with the system are very nicely done and contain all the information a user (or programmer!) would want.

I know I've left out some of the things that the PKASO /// system can do, but some of these are very specialized and the average user will never use them. Suffice it to say that the PKASO /// system is very, very powerfull.

Well, we had to come to it sometime. Yes, this package does have some bad points! Even though the PKASO /// command language greatly simplifies printer operations, it is not used by many application programs directly. Rather, the user must still type in character sequences (albeit somewhat shorter) to change text options and so forth. I'm saying this so that no one will get the impression that the PKASO /// is a cure-all. It's certainly not that, but it does come close!

The only other problem is not really a problem at all but just a general nuisance. The interface card that comes with the PKASO /// fills about ¼ of the space for one card in the peripheral card well. With all the 'real-estate' available for an interface card you would think they could have put some more things on the card. It's a shame to waste the space considering we only have four slots.

Figure #4

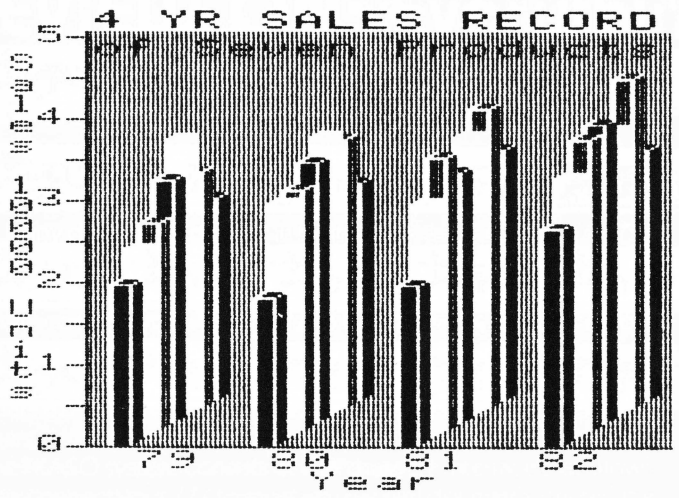
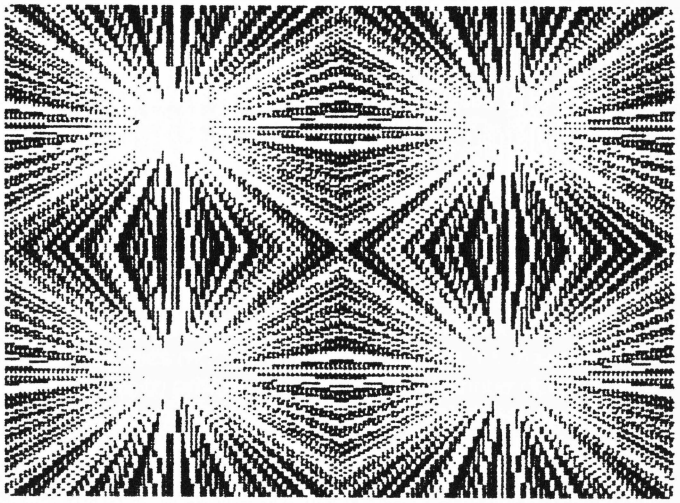


Figure # 5



Like I said, it's not really a problem. Though, it would have been nice to see at least another parallel interface port on the same card. Oh well, I'm just dreaming again. Maybe the next version! For now it's time for the . . .

Summary

From what has been said, the PKASO /// interface is a very good product in both quality and ease of use, but should you buy it? If you have any of the dot matrix printers that this card works with, I'd say an emphatic **YES!** For the money and for what it does, this is the card to buy. Interactive Structures, maker of the PKASO /// system should be given a round of applause for a very nice Apple /// interface card.

Items used in this review:

128K Apple ///
1 external floppy drive
MX-100 printer

Interface card: PKASO /// interface for the Apple ///
Version: ROM - 6.1; Manual - October, 1982;
Driver - Revision 3
Contents: Interface card, two instruction manuals, two program diskettes.
Programming language: Assembly, Basic and Pascal
Operating System: Standard SOS for Apple /// native mode, DOS for emulation.
Copy Protected: No
Warranty: 90 days
Cost: $205.00

The Bottom Line

PKASO /// system

Performance: Excellent
Documentation: Excellent
Ease of Use: Excellent
Error Handling: Good-Excellent
Over All Rating: A

Quick & Easy Data Master

The advertising for this program says that it is **'A Data Base Program That YOU Can Master'.** Does this claim hold water? Well, after a careful review I have found that it all depends on who **YOU** are. If you don't mind (or like!) making a moderate amount of changes in the Basic program, it's for you. However, if you don't know how to program I'd suggest you look elsewhere.

Another part of the advertising for this program says something to the effect, 'The ideal data base program is one that you can design exactly the way you want it: prompts, edits, error messages . . . to your specifications.' This is true, I always want to make some changes in the applications programs I own. It is also true that this package lets you define all these things.

Quick & Easy Data Master is a set of Basic programs that allow you to modify a generic Basic Data Base Program into what you want. It accomplishes this through an interactive question and answer session which asks you exactly where you want to put prompts, error messages and other items on your screen.

When you finish answering the questions, a custom stand alone data base managment program is created. You can then RUN the program and use it. Unlike most other Data Base Managements packages, if there is something in the program that you don't like, you can change it.

Since this is an idealized situation, let's say right off that there are a few problems with this package. First of all, the documentation stinks. The instructions manual consists mostly of bad photo-copies. The average user can work around this problem, yet the information on the photcopies is also bad. This is a real shame because the program has many good points. If the documentation was as good, this would be a very useful program.

Enough of that, let's now look at what the program does. There are two program disks in the package. After you boot the system disk, a message will be displayed to insert the other disk in the built in drive. Next a menu screen will be displayed. The options are as follows:

D . . . Demo Quick & Easy
E . . . Edit a program
R . . . Run Quick & Easy
X . . . Exit Quick & Easy

The option 'Demo' allows you to generate a demonstration program from a pre-defined format. You can use this option to see exactly how a data base program is created by this package. The program disks come with two pre-defined formats that you can use with the demonstration. One is a simple mailing list format and the other is a full blown sales invoicing format.

If you choose the 'Demo' option, you will get first hand knowledge of how the application program was made. If you are going to use this package in any way, this option will help you out. You can have it run automatically from the pre-defined format or pause after each question. A very good learning tool for the beginner.

You can use the 'Edit' option after you have made your data base program and discover you need to make some changes. This is particularly useful because you don't have to re-type all of the previous program format. If you want to change the menu titles or somethings like that, this option makes it easy.

The 'Run' option lets you use the Quick & Easy Data Master to create your custom application program. After answering all the questions, your answers will be used to create the program. These answers will also be saved so that you may later change them with the 'Edit' option, or use with 'Demo'.

The last option is 'Exit'. You can use this option to leave the application program and go to Basic. From here you can RUN any of the programs you may have created or any other Basic programs you may have.

Since all the other options are based on it, we will now discuss the 'Run' option. As stated, this allows you to create your own custom data base management program. Before you start it up you must first design the screen displays that you will use in your application program. To do this you must fill in the worksheets provided with the system.

These worksheets are just pages with numbered grids corresponding to screen positions. You fill in what goes where and then read it off the page when the program prompts you for where to put items on the screen. This isn't too bad, but the worksheets seem to have been designed for a 40 column screen. A second page is provided for columns 40 through 80. This is unfortunate because it make it a little harder for the user. One other complaint I have is that only one sheet is provided, if you're not perfect you are going to have to make some copies. I'd suggest about twenty.

After you finish designing the overall format for your programs screens you should use the 'Run' option to start the creation of your program. Now you will be asked questions such as the name of the program, any title message you would like displayed by your application program and other general information.

The most important questions come next. Here is where you must have everything figured out on paper. The program will ask you the maximum amount of space that should be reserved for each field of information. You can't just think about something in your head and try to answer the questions, you have to figure it all on paper **FIRST.**

There is one annoying part of this, someone accidentally spelled 'field', 'feild'. Come on guys, you should of caught that mistake long before you released the program.

Anyway, after you specify how much information each field is supposed to hold, you are asked which field is to be the key item. This is the part of the record that is found fastest in a search, and some though must go into the decision. If you are creating a mailing list that will be sorted by zip code, the zip code should be the key item. However, if you need quick retrieval in another field, you should choose that one as the key item.

You can not set up a double key, whereby you can get to any record by either, say name or zip code very fast. I sometimes wonder if the people who write data base management programs ever finished high school let alone get a degree. Doesn't anyone else out there know how to set up a file handler with a double key?

Getting back to business, after you choose a key field you must then say exactly where on the output screen to put all the items in the record. Next, enter all the prompts that are associated with each field in the record. Now you must indicate what type of information is to be entered into each field. Your choices are numeric or character data. A zip code and dollar amounts are both numeric types of information while a name or address is character.

One of the nicest features of the program is the ability to have your application program detect bad information when it is typed in. You can specify that it will be bad input if it in a date format, not alphabetic, not a state or not an entry (blank).

You can cause bad input messages to be displayed if the user types in any kind of data that you don't approve of. Thus, if a character entered is not numeric; not found in a specific file, an error message can be generated.

This is a very nice featue that I wish more programs would offer. After all this information is entered, you must then answer whether or not you want additional prompts or headers on the screen in addition to what was defined in the prompt specifications.

The next round of questions you must answer is whether or not you need to perform calculations on any of the fields in the record. Say you want to take 6% sales tax off of one line and put it on another, this procedure will allow that.

In addition to computed data, this package allows you to share data from one program to another by using interactive files. This allows you to transfer information from one file to another automatically.

We're done! After quite a few minutes of program design you now will have a complete data base management program written in Basic that you can make small changes in or major ones if you really don't like the way the program is written.

Now is time to look into the applications programs that it creates from your instructions. When you RUN the program it will ask you what data file to use in the program. If the file doesn't exist you can initialize the new file and then use it. When finished finding or initializing the data file, you will get a menu of all the available options. They are reprinted below:

DATA ENTRY

E — Enter Data
L — Lookup Single Records
U — Updata Single Records
D — Delete Single Records
F — Update Any/All Records
S — Scan/Modify All Recordrs
X — Exit Program
I — Initialize/Fix Data File

REPORTING

M — Make a Report
P — Print a Report

Once you get to this menu you can now choose any of the above options. Since there are quite a few, let me just tell of some interesting things I noticed with the created programs.

When you are updating a record in your data base, you can't change the key field, it's not allowed. Thus, if the person changed his or her name, this program would be in trouble. The lookup function is fairly standard, you can search for items using wildcards that will match only certain things in each item.

I've had some intermittent problems with the 'Scan/Modify' option. At times it will just not work. It seems that you must first attempt and fail to use this option a few times before it does work.

You can generate custom reports and save them using the 'Make a Report' option of the main menu. You can put the information from a data screen into any print format you desire and have the computer print your data out in that format. You may also save the

report specification to disk for later use.

I have found another intermittent problem with this option. It seems that after creating a report specification and writing it on disk, the file is not closed. Because the application program is written in Basic, the user can change any part of the program that they don't like.

When printing a custom report I have found another serious problem with this package. When printing to a disk file, only one record is ever written! It seems that the records are printed with an absolute record number and thus the only one you will ever see is the last one printed. This is only a problem when printing to a text file on disk. Printing to the console or printer is fine.

For those few who want to make some changes to the program, four pages in the manual are devoted to technical information of the program. The major program variables are explained and the major routines in the program are documented. Even the B-Tree file handler (and how to use it) is explained.

Below are some timings I made of the programs performance. Beware, they will change depending on the number of active drivers, and the size and speed of any disk drives you may have in your system. I used a standard Apple /// floppy in my tests. Hard disk drives (of course) will give much quicker operations.

Boot Time: 22 seconds with given drivers.
Print 100 labels to disk: (below design) 5:20.

Name: _____

Address: _____

City: _____ State: _____ Zip: _____

Disk Capacity: (above design): Approx. 950.

Summary

With a little better documentation and quite a few minor program changes this would be a very useful data base management program. However, if you don't have the knowledge or time to make those changes - it's not for you. As I see it, this package is a good buy for people willing to play around with a program that is generally good, but needs some work.

Advanced Software Technologym Inc. is to be commended with leaving their programs 'Open' so that people can tinker with them. Very few other manufactures will do this and for mostly that reason (the price is good too!) I find that this program is worthy of our support, even if it does have some major problems. ///

Equipment used in the review:

128K Apple ///
1 external floppy drive

Program: Quick & Easy Data Master for the Apple ///
Version: Program created 3/4/82

Contents: Two program diskettes, User's manual
Programming language: Basic
Operating System: Standard SOS
Copy Protected: No
Disk Warranty: None
Cost: $69.95

The Bottom Line

Quick & Easy Data Master

Performance: Poor-Fair
Documentation: Poor
Ease of Use: Fair-Good
Error of Handling: Good
Over All Rating: C

ProFile & Backup ///

ProFile

The ProFile drive for the Apple /// is a fixed-media random access 5¼ inch disk drive and disk controller card with a capacity of 5 megabytes (that's 5 million bytes!). That's a description of Apple's hard disk. For a little closer look at this device, read on.

If your data files feel a bit cramped using the standard old 140K floppies or if you just need faster disk operations, a hard disk may be for you. The ProFile will store the equivalent of 35 floppy disks and access the information about ten times the speed of a floppy.

This sounds great, right? Well, there are some drawbacks. First, unlike a floppy disk drive, you can't remove the disks. That is what 'fixed-media' means. Thus you are stuck with the disks that are sealed in the drive. The second problem is that it's only 5 megabytes. I know that it may seem like a lot, but after you begin putting all your software and data files on it you will see that you may want more room. The last major problem is that the interface card that connects your /// to the ProFile only can attach to one ProFile.

The first two problems aren't that great, but the last one is. Since the Apple /// has **only four slots** we must be careful how we fill them. Other disk interface cards allow you to hook up to a total of four disk drives, and I find that using up a whole slot for just one disk drive is just plain crazy. There is more than enough room on the ProFile interface card to do this and I can't see why Apple didn't at least put two interfaces on the card.

Oh well, you can't have everything. The ProFile is a very well built and reliable disk drive. I understand that to date not one has been returned to Apple due to a hardware failure. Now that's a very good number! I've had mine for about 17 months now and have yet to loose any data.

One of the things that makes the ProFile so reliable is that whenever you turn it on, it goes through a self-test procedure that does a complete surface analysis of the disk, checking for any marginal areas. If a bad sector is found it is automatically spared out to one of the sectors that is kept off-line for these purposes.

What's all this stuff about 'spared-out'? Well, the ProFile disk drive has a storage capacity of 9792 blocks, or a little over 5 million bytes of information. The average user will see somewhat less than that however. 64 of those 9792 blocks are reserved (or spared). When it performs the self-test these are the blocks that are used if a bad block is found on the disk.

Even though I have not lost any data, I have encountered bad blocks on the ProFile disk drive. These have always given some strange disk I/O error message. After getting these bad blocks I have been able to recover by simply turning off the drive and then turning it back on again. When it went through the self-test it found the bad blocks and replaced them with fresh new ones.

This is a very nice feature that I don't think any other drive offers. It is well worth the extra 60 seconds or so it adds to the time it takes before the disk can be used. Now all your programs can use larger

data files than what are allowed on the standard 140K floppy.

The instruction manual that comes with the system shows you exactly how to install the disk interface card and add the ProFile disk driver to your boot diskettes. If you don't want to, your dealer will install the interface and the necessary driver, but the manual does give you all the details. It also devotes a chapter to how the drive operates. I don't think that any user would ever need the information in that chapter, but if you want to know exactly how the drive works, there's a chapter for you.

There is also an appendix on how to add Pascal system files to the ProFile. This is not a very sophisticated method and you would be better off if you added Pascal to the ProFile using **Quark's Catalyst or Dr. Jeppson's Pascal Patch.**

One of the nicest features of the ProFile is the demonstration program that comes with the system. A color graphics demo second to none! Well over one hundred pictures are stored on the ProFile and are displayed by the controlling program. It's a real sight to see!

Backup ///

One of the strong and weak points of any large capacity disk drive is its ability to store very large files. A strong point because you can make larger files than with a floppy disk. A weak point because you can't make a copy of a 2000 block file on a 280 block floppy disk.

The program Backup /// was created to solve this problem. Very simply put, the program allows you to copy information from one storage device (a ProFile) onto a number of disks in another drive (a Disk ///) and later restore the information if needed.

This means that you can copy any ProFile file too large to fit on one diskette onto several diskettes. You can even do a full volume backup and store everything on the ProFile to floppy disks. A variety of options also let you backup and restore selected files to and from the ProFile. A very sophisticated and easy to use program, it is now supplied with each new ProFile that is sold, and has been sent to some current ProFile owners.

I said some because I have received many calls and letters from people who say they bought their ProFile months and months ago and have yet to receive their copies of Backup ///. I wish I could say that the reason that they didn't get their copy was because they didn't send in their registration card, but I can't. One gentlemen who phoned me said that he bought his ProFile in June of last year and promptly sent in his registration card, but has yet to receive Backup ///. After many calls and letters to both his dealer and Apple he still has not gotten his copy.

If you have ProFile and do not have a copy of Backup /// and your dealer can't get one for you, address your complaints to the following person:

Clayta Morand
Peripheral Systems Marketing
Apple Computer, Inc.
2720 Orchard Parkway
San Jose, California 95134

After spending a couple of thousand dollars on a ProFile, Apple should be a little more responsive to these problems.

The User's manual that comes with Backup /// is one of the best that Apple has ever done. Not just a reference booklet on how to use the program, it is a complete tutorial that shows you exactly how to backup and restore your files.

Backup /// will also format disks so you don't have to first boot the System Utilities Program and format the disks there. When backing up files you won't be caught without a useable diskette because with this program you can just take a blank disk just out of the box and let the program format it.

Backing up files couldn't be simpler, you just specify which files you want to use and insert the disks when prompted to. The program first verifies that the disk is good and then begins the copying. When finished putting information on that disk, it again verifies the disk to give an extra level of assurance that your backup disks are good.

There are three options for backing up your files. You can backup by Pathname, backup Modified files and backup by Date/Time. With these options you can specify many different file combinations to backup by.

The backup by Pathname option lets you backup an entire volume or part of a volume. You can use a device name, pathname, or a file pattern. The wildcards used in the file pattern are the same as used by the System Utilities Program.

The backup Modified files option lets you backup files that have been modified since you last made a backup. Thus if you have added or changed files on your hard disk since the time you last backed it up, you can use this option to copy only the files that have been changed. This is very useful if you have changed quite a few files on a number of subdirectories and can't remember where all of them are.

The backup by Date/Time option lets you backup files based on the date and time of last modification. Thus you could specify to make a backup of only files that have been modified since a certain date. This is useful only if your Apple /// has a functioning clock. Otherwise you must have set the date and time with the System Utilities Program for it to be of any use.

Restoring files is just as easy. Just specify which files you want to restore and insert the proper disks when prompted. Very, very simple! Towards the end of the manual is a section devoted to examples of using Backup /// to backup and restore files. Just another nice feature of this manual.

Three appendices cover everything from hints to a list of commands and even a complete section on error messages and how to recover from errors. It would have been better if these were reproduced on a handy reference card, but just being there is very good.

If you have another disk drive, such as one of the **Micro-Sci** high density disks, you can use Backup /// to backup files with greater ease. Using the **Micro-Sci A143** disk drive that holds four times the amount of a normal disk drive **(over half a megabyte!)** you can back up an entire ProFile with 10 disks or less! While a full volume backup of the ProFile onto standard 140K disk drives takes about 90 minutes, using the **Micro-Sci A143** it only takes about 30 minutes.

Summary

The ProFile - Backup /// combination is a fantastic tool for all Apple /// users. With the speed and capacity of the ProFile and the backup abilities of Backup /// I highly recommend these items to all Apple /// users who find that the Disk /// is just too limited. With the addition of a high density disk drive by **Micro-Sci,** this tool is further enhanced.

Equipment used in the review:

128K Apple ///
1 ProFile hard disk drive
1 external floppy drive

Item: ProFile Hard Disk Drive & ProFile driver
Version: Driver - Version 1.30
Contents: Interface card, ProFile Hard Disk
Owner's Manual, two program diskettes.
Operating System: Standard SOS
Warranty: 90 days
Cost: $2199

Item: Backup ///
Version: 1.0
Contents: Backup /// User's Manual, Program Diskette
Programming language: Pascal
Operating System: Standard SOS
Copy Protected: No
Disk Warranty: 90 days
Cost: Included with ProFile Disk

The Bottom Line

ProFile - Backup /// System

Performance: Good-Excellent
Documentation: Excellent
Error Handling: Excellent
Over All Rating: A

CRITICAL PATH SCHEDULING

If you are responsible for project scheduling, you know about the hours of planning and re-planning involved in the execution of a complex project. With the computer age upon us, it was just a matter of time before project management could be simplified. The CRITICAL PATH SCHEDULING system by Great Divide Software is the management tool that does just that.

I am going to assume that the reader knows a little about critical path and stuff of that nature. A detailed explanation is beyond the scope of this review so it will be left out. This review will show if the system is easy to use, and any problems (or strong points) that the package has.

To start off, the CRITICAL PATH SCHEDULING system is a collection of Basic programs that provides project managers a tool for defining and analyzing the overall concepts of a project and provides a powerful method for scheduling the many tasks necessary to complete the project.

Using the CRITICAL PATH SCHEDULING system, it is easy to insure that all important tasks are included at the start of the project. Using the information that the user furnishes for each task, the program analyzes the whole project from start to finish. A report can then be compiled that lists every task, with start and completion times. Also shown on the report are the tasks that are most critical to the completion of the project.

Each task generates an early start and finish date, and a late start and finish date. These dates are the earliest a task can be expected to start and end, and the latest the task can be postponed without affecting the completion time of the project. The difference between these two dates is called the 'float' and is included on the report. Also included is the 'free float' which is the amount of time a task can be delayed without affecting any other task or the completion date of the project.

The project reports can be in a variety of formats. Reports by early or late start, node or float order are all possible. Special formats such as five, six or seven day work weeks are a snap to prepare. You can also specify up to 100 non-working days in any project. Another very nice feature is the ability to do a report without specifying a start date for the project. Thus you can determine the length of time a project will take, regardless of the starting date.

The package comes complete with a detailed user guide/instruction manual and two program diskettes. Since the creators of this program rightfully realized that hard disks will give much quicker program operations, they included the capability of putting the CRITICAL PATH SCHEDULING system onto your hard disk.

When you first boot this system, a system configuration menu appears. Simply put, you have the option of putting all of the programs and data on the hard disk, programs on diskette - data on hard disk, or both programs and data on diskette. You can later change this if you wish. Thus you are not forever stuck with your CRITICAL PATH SCHEDULING system on diskettes or hard disk.

If you decide to put the programs on your hard disk, they are all moved automatically. This process takes about four minutes with the ProFile. You can manually turn the screen off ('CONTROL 5') to speed this up a bit.

The boot diskette that comes with the system has the '.PROFILE' hard disk driver on it, so you won't have to put any overtime in with the System Utilities Disk. You aren't locked in to just the ProFile either. Where the programs are put is entirely up to you. If you want to use a hard disk other than a ProFile, you will have to add its driver to the boot disk's SOS.DRIVER file.

The ability to transfer programs to a hard disk is seldom found in the computer programs of today and I applaud the people at Great Divide Software for integrating their software in this manner.

The CRITICAL PATH SCHEDULING system allows multiple projects to be on a hard disk at once. It also allows you to switch between projects very quickly. With the programs on the hard disk, a menu of current projects is displayed upon booting. Choose the one you want and you get it. If you want to change from one to another you don't have to reboot the system, just choose one of the menu options and you can go to a different project.

One of the nice things about the CRITICAL PATH SCHEDULING system is that it comes with a set of sample data. This allows you to practice using the system before you set up your own projects. With this you don't have to worry about hurting your own data!

After choosing the project you want to work on, the main menu is displayed. It is reproduced here as figure #1.

Figure #1

CRITICAL PATH SCHEDULING SYSTEM MENU
SAMPLE PROJECT

→0 QUIT

→1 ENTER/ACCESS PROJ.INFO
→2 ENTER TASK DATA
→3 ACCESS TASK DATA
→4 PRINT TASK DATA

5 SORT AND TEST DATA
6 CALCULATE CRITICAL PATH

→7 SWITCH PROJECTS
→8 MOVE FORMATTED REPORTS
→9 CREATE/COPY/DELETE A PROJECT

BASIC REPORTS
→10 CRITICAL PATH REPORT
→11 KEY WORD LIST
→12 MANPOWER REPORT
→13 TASK BAR CHART

DATED REPORTS
→14 CRITICAL PATH REPORT
→15 KEY WORD LIST
→16 MANPOWER REPORT
→17 TASK BAR CHART

→18 SPOOL REPORTS

DESIRED OPTION NO. ?

As you can see, arrows point to some of the options. These are used to indicate which options are currently available for use. The above example shows that we are using the 'SAMPLE' project that is included with the system. Since there is data, there are arrows pointing to options 0-4 and 7-18. These are the ones we can presently use.

One feature of the system that I don't like is option #0. This is the 'QUIT' option and if you press 0 and RETURN, you can not recover the program. There should be some prompt message asking if the user really wants to quit. Over the past couple of months I have accidentally chosen option #0 many times.

The instruction manual that comes with the CRITICAL PATH SCHEDULING system is very good. It contains step by step instructions that leads the user through the system. While not typeset, the instruction manual comes in three ring binder format. One of the biggest reasons that I like the manual is because the pages are printed on only one side. Thus there are no readability problems due to seeing the back side of a page through the front.

After preparing your project's data using the example in the instruction manual, you enter the project information using option #1 of the main menu. Here you tell the program the name of the project, the starting month, day and year. Also included is the work week type (5,6 or 7 days), and the non-working days during the project.

Here you will also specify where to send any report, in other words - which printer driver. This will normally be '.PRINTER', but the user can change it to any configured printer driver. You can also specify any control characters to send before printing the report. This enables you to force compressed printing to squeeze more information on a page.

A report heading can also be specified which will appear at the top of all printed reports. This can be up to 60 characters long. To maximize memory use in the Apple /// this option also asks you to specify the maximum number of tasks your project will need. Up to 2150 can be specified, but it's best to use the default value (1000) unless you know you will need more.

The next three options (number's 2-4) are used to enter, update and print the task data. These processes are very simple due to good instructions and screen menus. Complete editing functions are included so the user will not have to re-type the entire task data due to a single mistake.

Option number 5 sorts and tests the task data. It tests to check that the data is correct before you compute the critical path. You can also have the program directly go to option number 6 and compute the critical path if the data is correct. Any errors are displayed on the screen for the user to see and later correct.

If you have finished with your work on one project and want to go to another, option number 7 allows you to switch projects without rebooting the system. Each project's task information is kept on a different set of data disks, or on different subdirectories if you are using a hard disk.

Option number 8 is a bit confusing. It allows you to view report files stored on disk, move the report files to a printer, and delete report files to regain disk space. Now what is a report file? Well, that's not so hard. They are CRITICAL PATH SCHEDULING reports that are saved on a disk instead of being printed on a printer. Now, option number 8 allows you to manipulate these reports, but how do you create them? There isn't anything in the table of contents!

After much searching (two days worth) I finally found out how to do it. Directly after the description of option number 4 there are two special notes. The first of these is how to get a report printed on the screen. After using the system for a while I had found this out. All you do is press 'OPEN APPLE S' and the report will be printed on the screen. I learned this because on the report options menus there is a line that says to press 'OPEN APPLE S' to get a screen print-out.

The second of these special notes is how to print the reports to a disk file. Instead of pressing 'OPEN APPLE S', simply press 'OPEN APPLE F' and you will be prompted for a File to send the report to. Why doesn't this merit a special line like the print to screen line? I don't know! There should at least be a section in the table of contents on how to accomplish it. Hopefully the people at Great Divide Software will fix this little bug before it gives anyone else headaches.

While we are on the subject, about the only other problem in the manual is on pages 34 and 35. On these pages it states that there are two formatted reports on the diskette that comes with the system. In the manual they are called SAMPLE1 & SAMPLE2, yet they are not on the diskettes that I received.

Option number 9 allows the user to create, copy and delete new projects. This is very useful, because the user doesn't have to boot up the System Utilities Disk to make a copy of their important task data.

Options 10 through 18 are used to print reports in a variety of options. Reports using early and late start dates, float and node order are easily printed. One of the nice features is the ability to print out a key word report. With this capability you can print a selected portion of the critical path report based on a 'key word' in the description of the tasks.

Thus you can specify work that is say, subcontracted out by putting the word 'Sub' in the description of each task and later print out only that portion of the report whose tasks descriptions begin with 'Sub'. A very useful feature.

Manpower reports that indicate the total manpower required for each task can be printed, giving the project manager total control in the determination of crew size.

Probably the most important report option is the 'Task Bar Chart'. This report gives the same basic information as the other reports, but gives it in a pseudo-graphical format. In this way, the user can see how various tasks relate to each other. Text symbols (*=) are used to show critical and non-critical tasks in the project.

This is a very nice feature because it shows the user exactly which tasks can be delayed, and when they must be completed. Though nice, I think it should be done a little differently. With the exciting graphics capability that the Apple /// offers, a much better implementation should have been written.

On each report you can have the current date printed. The problem here is that you must enter the date manually. Since many Apple /// owners now have a clock installed, programmers should use the clock instead of always asking the user to supply the date.

Summary

The CRITICAL PATH SCHEDULING system is a very useful tool for anyone who is involved with project planning. It isn't a Lisa Project, yet it does provide a very useful function for project managers.

The few minor problems in the program and documentation are hardly enough to turn me away from the program, but the price $495 makes me think twice. Because it is written in Business Basic, the more advanced user is able to tinker with the system to adjust it more to his or her needs. Since it does have many fine features, I think it deserves our support.

Overall the CRITICAL PATH SCHEDULING package does help project managers do their job, so this may be the package for you.

Continued on page 9

Three Shorts — Fini!

by Devin Sexson

The following are snappy little graphics exercises with levels of complexity and thoughtfulness that are unusual.

To use, just type them in and save them. Make sure the Basic disk is on-line and type "RUN." The Basic disk is needed because the file "BGRAF.INV" is used.

ON THREE will pay $25 for any short demonstration program used in this space, so send in your favorite today, and we will see you next time on ON THREE. ///

```
0   REM ****************************************
1   REM *  Dazzling Rectangles                 *
2   REM *  -------------------                 *
3   REM *  This short program displays a neat  *
4   REM *  pattern of dazzling rectangles in full *
5   REM *  color.  To use, type in the program  *
6   REM *  and then 'RUN'.  Make sure you have  *
7   REM *  the '/BASIC/BGRAF.INV' on-line.  If  *
8   REM *  not, enter the correct pathname.     *
9   REM ****************************************
10   ON ERR INVOKE"/BASIC/BGRAF.INV"
20   PERFORM initgrafix:OFF ERR
30   PERFORM grafixmode(%1,%1)
40   PERFORM grafixon
50   PERFORM fillport
60   i%=3:x%=0:z%=279:q%=0:y%=191
70   PERFORM viewport(%x%,%z%,%q%,%y%)
99   ON KBD GOTO 1000
100   FOR r%=1 TO 99
110     pc%=pc%+1:IF pc%=16 THEN pc%=1
120     PERFORM pencolor(%pc%)
130     PERFORM moveto(%x%,%y%)
140     PERFORM lineto(%z%,%y%)
150     PERFORM lineto(%z%,%q%)
160     PERFORM lineto(%x%,%q%)
170     PERFORM lineto(%x%,%y%)
180     x%=x%+i%:y%=y%-i%:z%=z%-i%:q%=q%+i%
190     NEXT r%
200   TEXT:END
1000   IF KBD=27 THEN POP:TEXT:HOME:END
1010   ON KBD GOTO 1000
1020   RETURN
```

```
0   REM ****************************************
1   REM *  Random Triangles                    *
2   REM *  ----------------                    *
3   REM *  This program will display a number of  *
4   REM *  tringles of random size, in a variety  *
5   REM *  of colors.  To use, enter the program  *
6   REM *  and then 'RUN'.  Make sure that the    *
7   REM *  file '/BASIC/BGRAF.INV' is on-line.    *
8   REM *  If not, enter the correct pathname.    *
9   REM ****************************************
10   ON ERR INVOKE"/BGRAF.INV"
20   PERFORM initgrafix:OFF ERR
30   PERFORM grafixmode(%3,%1)
40   PERFORM grafixon
50   PERFORM fillport
60   PERFORM viewport(%0,%139,%0,%191)
99   ON KBD GOTO 1000
100   FOR r=1 TO 20
110     x1%=139*RND(1):y1%=191*RND(1)
120     x2%=139*RND(1):y2%=191*RND(1)
130     x3%=139*RND(1):y3%=191*RND(1)
140     PERFORM pencolor(%RND(1)*16)
150     PERFORM moveto(%x1%,%y1%)
160     PERFORM lineto(%x2%,%y2%)
170     PERFORM lineto(%x3%,%y3%)
180     PERFORM lineto(%x1%,%y1%)
190     NEXT
200   x1%=139*RND(1):y1%=191*RND(1)
210   x2%=139*RND(1):y2%=191*RND(1)
220   PERFORM viewport(%x1%,%y1%,%x2%,%y2%)
230   PERFORM fillport
240   PERFORM viewport(%0,%139,%0,%191)
250   GOTO 100
999   TEXT:END
1000   IF KBD=27 THEN POP:TEXT:END
1010   ON KBD GOTO 1000
1020   RETURN
```

```
0   REM ****************************************
1   REM *  Moving Color In A Diamond           *
2   REM *  ------------------------            *
3   REM *  This program will display a pattern  *
4   REM *  consisting of a colorful diamond with  *
5   REM *  colors constantly changing within it.  *
6   REM *  To use, type in the program and then   *
7   REM *  'RUN'.  Make sure that you have the    *
8   REM *  file '/BASIC/BGRAF.INV' on-line.       *
9   REM ****************************************
10   ON ERR INVOKE"/BGRAF.INV"
20   PERFORM initgrafix:OFF ERR
30   PERFORM grafixmode(%1,%1)
40   PERFORM grafixon
50   PERFORM fillcolor(%0)
60   PERFORM fillport
99   ON KBD GOTO 1000
100   z%=1:g%=280:h%=96:a%=140:c%=236
110   d%=140:e%=-44:i%=0:j%=96
120   FOR k%=236 TO 97 STEP-1
130     f%=16*RND(1):PERFORM pencolor(%f%)
140     PERFORM moveto(%a%,%c%)
150     PERFORM lineto(%g%,%h%)
160     PERFORM lineto(%d%,%e%)
170     PERFORM lineto(%i%,%j%)
180     PERFORM lineto(%a%,%c%)
190     g%=g%-z%:h%=h%-z%:d%=d%+z%:e%=e%+z%
200     i%=i%+z%:j%=j%+z%:a%=a%-z%:c%=c%-z%
210     NEXT k%
300   TEXT:END
1000   IF KBD=27 THEN POP:TEXT:HOME:END
1010   ON KBD GOTO 1000
1020   RETURN
```

Lazarus /// (Undelete your deleted files!)

How much are your important data files worth? $100, $1000? Even if you back up your files regularly, the one file you accidentally delete will be the one you haven't ever backed up.

Wouldn't it be great if you could somehow regain those files you deleted? Well, with **Lazarus** /// - you can! Very easy to use, just insert the diskette with the files you deleted and **Lazarus** /// will recover it. Completely user friendly, this program has on-line help and tutorial screens to aid in the use of the program. It even works with ProFile and other disk drives!

If you order this program today, you can get it for the pre-introductory price of $24.95. Orders postmarked after July **30**, 1983 will be sold at the full price of $29.95. This program will be shipped on July **30**, so place your order today for the best Apple /// utility in town. Please add $1.50 for shipping and handling.

Disk Of the Month

Do you have the time to type in the programs in each issue of ON THREE? Wouldn't it be great if there was a way to get all the programs without having to type them in? - There is, all you have to do is buy the disk!

DOM #1 - Extra Disk Space Plus!

This disk contains all the programs contained in the January and February-March issues of ON THREE. Included are Disk Pak1, which will give you four extra blocks of disk space on all your data disks (a very handy feature for those budget conscious people who don't have a hard disk!); Disk Pak2, which lists the files on a directory using Pascal; all of the Graphics and Sound Demos and much, much more!

DOM #2 - Changing the Characters Of Your Printer

This disk contains a program that will do a most amazing thing, it will enable you to change the characters that your Apple Dot Matrix (or Prowriter) printer prints with. Now your DMP can print with the same characters that are shown on your text screen. Fancy Gothic letters and many other fonts are now available to use on your printer. Complete documentation makes this program very easy to use. Also included on this disk is a program to list the files on an Apple][DOS diskette and many more graphic demonstrations.

For only $9.95 (plus $1.50 for postage and handling) you can get either of these great packages. If you want to order both you can get them for the extra low price of $15.00 (plus $2.00 for postage and handling). Order today!

Group rates are as follows:

2-9 disks: **$7.50** apiece + $2 total shipping
10-24 disks: **$7.00** apiece + $3 total shipping
over 24 disks: **$6.50** apiece + $4 total shipping

Group rates must have one mailing address. Please use the attached envelope for orders. If the envelope is missing, send to:

ON THREE
Attn: ORDER DEPT.
P.O. Box 3825
Ventura, California 93006

ON THREE O'Clock

Calling all you time conscious Apple /// owners out there. How would you like a working clock/calendar for your Apple ///? Just as it was originally intended, this kit comes complete with a plug in clock chip with a battery backup.

With ON THREE O'Clock installed, any time you save or modify a file, the current time and date will be stored on disk. Thus you will now be able to tell which file you last worked on. Your programs can now use the Apple /// built-in date and time routines to give you an up to the second read-out of what time it is.

Extremely easy to install and adjust, it is completely compatible with SOS and doesn't use up a slot! This is the one you have been waiting for! The package contains comprehensive instructions and a Six Month Warranty! Try to get that deal anywhere else!

What's the best part? - The price! While others are selling theirs for $60 and up, we have broken the $50 barrier. Heck, we broke the $40 barrier!

For only $39.95 (plus $2.50 for postage and handling) you can get the best little clock in town!

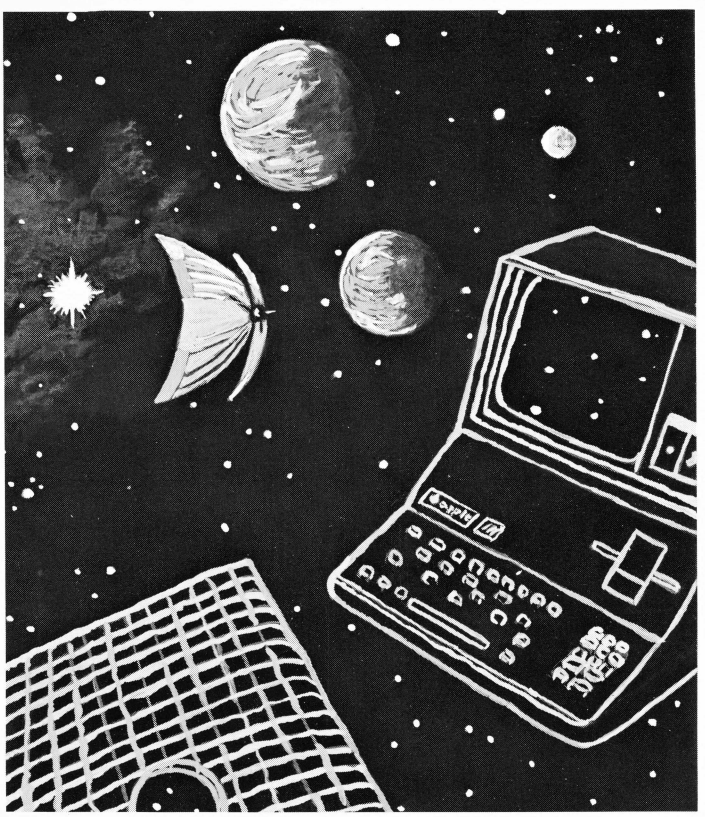
Group rates are as follows:

2-9 clock sets: **$36.50** apiece + $5 total shipping
10-24 clock sets: **$33.25** apiece + $7 total shipping
over 24 clock sets: **$31.00** apiece + $9 total shipping

Group rates must have one mailing address. Please use the attached envelope for orders. If the envelope is missing, send to:

ON THREE
Attn: ORDER DEPT.
P.O. Box 3825
Ventura, California 93006

For prices that are out of this world...



Buy ON THREE products!